A Report On Project
Entitled

AIR QUALITY MODELLING

Developed For
Tata Energy Research Institute (TERI)
9 Jor Bagh, New Delhi - 110 003

By

Pratikshit Mehta

For The Degree Of
Master Of Science

in

Mathematics With Computer Science, 1992

Department Of Mathematics
Jamia Millia Islamia
New Delhi

TATA ENERGY RESEARCH INSTITUTE

9, JOR BAGH
NEW DELHI-110003

CERTIFICATE

This is to certify that *Mr. Pratikshit Mehta*, a student of *M.Sc. Mathematics with Computer Science, Jamia Millia Islamia, New Delhi*, has undergone training at *Tata Energy Research Institute (TERI)*, and completed the project entitled "*Air Quality Modelling*" under my supervision. This report is submitted by Mr. Pratikshit Mehta in partial fulfilment for the award of Degree of M.Sc. at Jamia Millia Islamia, New Delhi.

S Saksena
(Mr. SUMEET SAKSENA)

GRAM : TERINST 110003—TELEX : 31-61593 TERI IN—FAX : 91-11-4621770—PHONES : 617025, 690503, 615032, 4621738, 4622246
Branch Offices : Tata Energy Research Institute "KRUPA" 50/7 Palace Road, Bangalore - 560052, INDIA
Tata Energy and Resources Institute, 1634, 19th Street, # 2, Washington, D.C. 20009, USA

DEPARTMENT OF MATHEMATICS
FACULTY OF NATURAL SCIENCES
JAMIA MILLIA ISLAMIA, NEW DELHI-25

Phone : 6844650
6831717
Ext. : 340
Gram : JAMIA

Date

C E R T I F I C A T E

This is to certify that **Mr. Pratikshit Mehta**, student of M.Sc. (Maths with Computer Science) has undergone training at Tata Energy Research Institute (TERI) from September 15, 1998 to February 15, 1999 and has completed the project entitled "AIR QUALITY MODELLING". The project report is being submitted for the partial fulfillment of the requirements for the award of M.Sc. (Maths with Computer Science).

The project report has not been submitted to any other institution except Jamia Millia Islamia to get any degree or certificate.

We wish him all the success in the future.

(Dr. Z.A. Taqvi)
Head.

TATA ENERGY RESEARCH INSTITUTE

M. Karuppasamy
Fellow
Computer Center

9, JOR BAGH
NEW DELHI-110 003

16th February, 1993

TO WHOMSOEVER IT MAY CONCERN

It is out of immense appreciation, I have written this letter of recommendation. Mr. Pratikshit Mehta was associated with me during his tenure as a trainee at Tata Energy Research Institute (TERI) from 16th September 1992 to 15th February 1993 as part of his Master in Science (Mathematics with computer science) final year project curriculum.

"Air Quality Modelling" software package was developed successfully by him using Borland's Turbo C++ development system. This model predicts approximate strengths of different sources of air pollutants at a particular site. During this exercise he got wide exposure to various mathematical, statistical and analytical techniques like Factor Analysis, Least Squares Method, etc. This software will be of great use in the project TERI has undertaken for United Nations Environment Programme (UNEP).

He is sincere and has an aptitude to work very hard. His programming and analytical skills are highly appreciated. I wish him all the success in his life.

(M. Karuppasamy)

Acknowledgement

No one can claim that he can do anything absolutely independently. But if someone helps you by all possible means, you just cannot go without expressing your gratitude. And I am not an exception.

I really find myself in debt of Dr. R.K. Pachauri, Director, Tata Energy Research Institute (TERI), New Delhi, for showing confidence in me by providing me a challenging job of developing this software package.

Certainly Dr. Z. A. Taqvi, Coordinator, Project & Examination, Lecturer Mr. Hamid Hassan and other faculty of Mathematics Department, Jamia Millia Islamia, are to be given full credit for the success of this project work. It has become possible only because of the knowledge which I gained under their able and expert guidance during my Master's Course.

I also wish to express my sincere gratitude to Mr. Sumeet Saksena, Research Associate, TERI, without whose excellent supervision I would not have been able to complete my project to its logical conclusion.

My thanks are also due to Mr. M. Karuppasamy, Fellow, Computer Center, TERI, who has always helped me out of the troubled moments in programming.

I would also like to express my sincere thanks to Ms. Sincy Joseph and Ms. Vidya for their secretarial help.

CONTENTS

CHAPTER 1

Introduction

Chapter 1

Introduction

Start with computers, and then computers, and then again computers, and at last computers. Yes, we have become so much dependent on computers that this 'machine' has become so integral part of our daily life that it has lost its identity. The morning telephone alarm, controlled by computerized electronic exchange, rings up to say " Good Morning ! ". So you are ready to leave for the office. On the counter, you are handed over a ticket printed out by a computer, for the local train service. In the office, the secretary gives a pleasant smile and then says, " Sorry ! I couldn't do it. The 'machine' is down since yesterday.", returning your papers. After having lunch, when you ask about the amount to be paid, the cashier pushes some buttons and your bill is ready. I think, that's enough ! Do I require to specify that computers are being used for rocket launching, architecturing, designing of machines & tools and...?

Research is no exception. Rather there is an extensive use of computers in diverse fields of research - the scientific research, the mathematical research, the social science research... - all have become much reliant on computers, and as a result, much advanced.

Pollution: A big threat to the nature and mankind. There are different kinds of pollution: air pollution, water pollution, noise pollution, and many others. Air pollution has become a big issue now a days. Air is getting polluted to the extent that breathing which is essential to survive, has become a curse. Can you say how much of particulates, SO_2, H_2S, C, CO,..., you have inhaled ever since you

started reading this text? What are the sources of all these invisible contributors? Yes! the research is to be paid credit which has made it possible to answer this question. Today we are having different methods and mathematical models to find the answers to these questions. There are different types of Source-Oriented & Receptor-Oriented methods existing. Here I have concentrated on the Receptor Oriented methods. Under these methods, we have Dispersion Models, Microscopic Methods, Chemical Methods, etc.

The Chemical Mass Balance (Elimination Method & Least-Squares Method) and Multivariate Methods (Factor Analysis) of Chemical Methods are of prime consideration here and are discussed briefly in chapter 4.

Tata Energy Research Institute (TERI) has gained international fame for its superb research in the fields of Non-Conventional Energy Sources, Environment, Biotechnology, development of models and software etc., in recent years. Chapter 2 gives a brief introduction to TERI.

Chapter 3 provides a briefing to C++, C++ graphics and differences between C++ and C. To understand C++ properly, preliminary knowledge of C is a must.

As already written, as a result of research, many mathematical models have come up. Manually applying the data to these models means heaps of calculations and weeks of time. This software is being developed to get the results by applying different models to the data. **FACTOR ANALYSIS** is of special mention here. This is a multivariate statistical technique. This is a young and very useful field of statistics. Factor analysis basically attempts to reduce huge amount of data into smaller, easily manageable factors. Few

packages, like, BMD, DATATEXT, OSIRIS, SAS, SPSS etc, are already existing which provide factor analysis solutions using different techniques.In this software, the basic factor solution is obtained by using Principal Factor Solution Method, and then Varimax and Quartimax rotations are applied to it to get the final solution. Chapter 5 give the mathematical concepts that are essential to be understood to know the different techniques.

Chapter 6 gives the complete logic of the software and that of the programs. Wherever possible, flowcharts are being provided to facilitate understanding.

Finally Appendix A; Programs Listing:- contains the listings of all the program files that can be referred to make renovations in the software as and when required. It is suggested to always keep the appendices upto date. Least to say, benefits shall be evident. Appendix B; References:- gives the details of the papers and books on which I have heavily relied upon, while developing the software.

CHAPTER 2

About TERI

Chapter 2

About TERI

TERI (Tata Energy Research Institute) was set up essentially to tackle the immense and acute problems that mankind is likely to face in the years ahead on account of (1) the gradual depletion of the earth's finite energy resources which are largely non-renewable; and (2) the existing methods of their use, which lead to increasing pollution.

The mission of TERI, therefore, is to find solutions to a range of problems; and it is this characteristic which imparts a definite goal orientation to all the activities undertaken within the Institute.

TERI's programmes reflect its wide-ranging charter and illustrate its integrated and multi-disciplinary approach. The Institute is currently active on eight broad fronts, namely (1) Energy Policy and Planning, (2) Rural Energy, (3) Forestry, (4) Renewable Energy Technologies (5) Energy conservation, (6) Bio technology, (7) Computer Applications, and (8) Training and Information Dissemination.

Programmes in each of these eight areas attack problems at several levels and scales. For instance, studies on smoke pollution in rural kitchens and studies in cell and protoplast cultures represent TERI's range at the micro level; and the development of mathematical models to depict and simulate the entire country's energy-economy system in 2001 and beyond is an example of work at the macro-level. TERI's training programmes are another example: they range from those aimed at the owners of improved *chulhas* and biogas plants to

those that involve policy makers and senior civil servants. Geographically to, TERI's brief is wide; it has been fortunate to receive the support of the Central and State Governments and of many international organizations including the World Bank, Ford Foundation, Rockefeller Foundation, International Development Research Center, Commission of the European Communities, the UN University, ESCAP, UNESCO, FAO and others. To complement its research and training activities, the Institute is active in publishing several forms of literature on energy and environment, ranging from abstracting services to in-depth state-of-the-art reports on various topics.

TERI was set up in 1974 with generous funding from some Tata companies spearheaded by Tata Chemicals. As TERI's stature is credibility have grown nationally and internationally, it continues to attract projects and support from a wide range of organizations, as it does talent to the ranks of its staff from the best institutions spread over the globe .

TERI is a young institute, growing rapidly. The professional vitality of its staff, a consciously built, simulating work-culture, and a sense of self-imposed discipline that blends the best of inspired management and a well regulated and a smoothly run administration are it's main features which make TERI an excellent organization.

TERI is now in its phase of consolidation during which it is building upon its current strengths. Most of its first Five Year Plan (1988-92) is now a reality; the second Five Year plan (1992-96) is now under way. TERI is widening its horizons further: the Institute now hosts the secretariat of Asian Energy Institute, a network of energy institutions from nearly a dozen of the Asia-pacific region. TERI is also one of the first developing country research organizations to

have set up in Washington -- the Tata Energy and Resources Institute was registered in the united States in August 1990.

Poised for the twenty-first century, TERI is confident of meeting the high expectations of its supporters and justifying the faith they have placed in it.

Computer center

The Computer Center of TERI is responsible for providing support and service for a wide variety of tasks that require applications of a computer. The Center conducts regular training programmes and user-group meetings where topics related to new and upgraded software and hardware environments, efficient use of resources and general problems are taken up. These also provide an opportunity for exchanging ideas and sharing experiences. In addition, researchers from this group participate in various software related to all activities in the Institute, the Center is faced with the challenge of rapid growth in terms of both hardware and software in the current and following years.

Hardware. The Institute is equipped with nearly 40 PC-compatible microsystems ranging from 386 AT to PCs, 20 printers including letter quality printers and laser printers, 35 mm color slide making machine, a CD-ROM drive and 2 data acquisition systems. Future acquisitions include a color plotter, more high-end PC/At systems and letter-quality printers.

Software. The Center's software library comprises DOS, UNIX and XENIX operating systems, various higher-level language compilers and word-processing, database, spreadsheet, graphics, public domain and application-specific software packages. This software library is

constantly upgraded to meet the growing needs of users and to keep pace with the technology. Recent additions include MS DOS 5.0 and CoDraw, technical drawing package.

ONGOING ACTIVITIES

TERINET. At present, the Center is actively involved in planning the computer set-up at the IHC (India Habitat Center), the 'TERINET'. Following aspects of the system are being worked out for TERINET: (1) likely load based on projected staff strength at IHC in 1995 and after, (2) requirements for stabilized power supply and power distribution network based on the projected distribution of computing load, (3) composition of hardware systems and peripherals to meet various needs, (4) composition of software packages and operating environments, (5) co-existence and resource sharing of PC network, DOS and UNIX systems, and (6) network applications such as E-mail, telex, fax remote access through modem, etc.

Tissue culture pilot plant software package. Development of Research Wing Database Module is complete. The second module, which deals with production scheduling, incorporates macro-level, and micro-level planning. Macro-level planning gives complete production layout for a particular species, while micro-level planning comprises daily, weekly and monthly plans for each state, namely growth room, hardening room nursery, and dispatch. System design for this module is finalized and program development is in progress.

COMPLETED ACTIVITIES

User interface for NO_x emission model. User interface in C was developed for this program which was written in FORTRAN.

F-value computation for nucleotide substitution. The F-value computing program in Pascal acquired by the Biotechnology group was modified and adapted to their requirements.

TERI Costing Model (TCM). A user friendly software package was developed to forecast TERI's future expenditure, based on past expenditure growth rate trends. It contains three modules, namely data updation, computation and report generation.

Pubtract. A software package was developed to keep computerized records of all components of publishing, namely production, order fulfillment, and marketing. The elements that form the basis of this management information system include subscribers' lists, details of mailing expenditure, records of printers' and paper merchants' bills, etc.

CHAPTER 3

About C++

Chapter 3

C++. Extensions to C

From C To C++

That's the logical second step. C++, an augmentation to C, finds the loopholes, in C, and fixes them. C++ is a lot tighter, not in terms of rigidity, but rather in terms of channelizing directions. C allowed to meander so much and so far, that direction often tended to go haywire. It allowed so many trials and errors, the temptation to deviate was irresistible. The glitches were many.. And the concentration and caution required of you as a conscientious programmer were phenomenal. C++ introduces small rules to the malleable masterplan of C. They are simple and convenient rules that don't take anything away from the sweet taste of freedom relished in C.

A program to initiate into the modified realm:

```
main ()
{
printf("Hello World\n");
}
```

You'd figure you'd get by without a hitch. But the C++ compiler throw back. An error. Before we check in to find out why, we'd like to say "Welcome to C++".

Now why was the error flagged?

The undemanding C structure would function unerringly to give you Hello World. C++, on the other hand, asks for very strong and conspicuous prototype checking.

C was primarily unconcerned with the logistics of passing variables to a function. But C++, in keeping with its role of the meticulous guide, demands clean and unambiguous definitions. It calls for, what are called, function prototypes.

Function Prototypes

A function prototype is a declaration that defines both, the parameters of the function and its return type. The prototype is checked against, each time the function is called, to ensure stability and unambiguity in code. The arguments passed in the called function are examined to determine whether these arguments correspond to the data types in the prototypes checklist.

The C++ compiler, thus, would understand the same program if you included the standard header file, STDIO.H, which contains the prototype of the function *printf()*, in the following manner:

```
#include <stdio.h>
main()
{
 printf("Hello world\n");
}
```

Consider this program.

```
#include <stdio.h>

fumbler(int foo,int beep)
{
 printf("%d...%d\n",foo,beep);
}
```

```
main()
{
 int boo = 10,bap = 20;
 fumbler(boo,bap);
}
```

It will not run because the function definition must include the return value as well. C took the unspecified return value as an *int* by default. C++ asks for the return value to be spelt out in so many words.

Parameter Lists And Return Values

Let's start with an example:

```
#include <stdio.h>
void fumbler(int foo,int beep)
{
 printf(%d...%d\n",foo,beep);
}
main()
{
 int boo=10,bap = 20;
 fumbler(boo,bap);
}
```

The prototypes have been affirmed, the return values have been specified, and the program will run without a flicker of doubt. The *printf()* outputted will be 10 and 20.

Now, take this program:

```
#include<stdio.h>
main()
{
  int boo=10,hoo=20;
  fumble(boo,hoo);
}
void fumbler(int foo,int beep)
{
  printf("%d...%d\n",foo,beep);
}
```

This program will again not work. That's because the prototype hasn't been specified initially. So *fumbler()*, when the prototype is not there to be checked against, does not mean anything to the C++ compiler. Hence, the error.

Which means if the definitions are enumerated later, the prototype should be defined initially.

Now let's get the program to work:

```
#include<stdio.h>

void fumbler(int,int);

main()
{
  int boo=10,bap=20;
  fumbler (boo,bap);
}
void fumbler(int foo,int beep)
{
```

```
  printf("%d...%d\n",foo,beep);
}
```

It will work perfectly. Because the prototype has been specified before the definition. The result outputted will be 10...20.

Let's go ahead with another program:

```
#include<stdio.h>

Void fumbler(int,int);

main()
{
  int boo=10,bap=20;
  fumbler(boo);
}
void fumbler(int foo,int beep)
{
  printf("%d...%d\n",foo,beep);
}
```

Error! While defining the prototype, it has been explicitly mentioned that it will accept two *ints*. C would have compiled this program and run it merrily. But C++ demands strong prototype checking. So, if you say you're passing it two parameters, you better stick to your words. C++ makes no excuses for shoddy programming.

Default Values

Here's a program that will work:

```
#include<stdio.h>

void fumbler(int foo=0,int beep=0)
{
  printf("%d...%d\n",foo,beep);
}
main()
{
  int boo=10,bap=20;

  fumbler(noo,bap);
  fumbler(boo);
  fumbler();
}
```

While defining the function prototype, we've also defined a default value for each parameter. Unlike C which takes any random value, C++ permits arrangements to be made for function parameters to take default values. Now, we can call the function with two or one or no parameters.

The output of the above program will be:

```
10...20
10...0
0...0
```

C++, in other words, builds a bridge between the user and the programmer. It function in the middle ground between two extremes - on the one hand, C with its complete indifference to what variables you pass and how many, and on the other, the extreme prototyping

tendency of C++ itself.

This middle ground provides a stand-by value, the default parameter. Allowing the user to state his parameters, in which case the default parameters are overwritten. But on those occasions when the user fails to define his own parameters, the default parameters bail him out. The program runs anyway, providing the freedom to choose one's own value, and if that choice is not taken, making the relevant provision.

Taking that further, when one parameter is passed, the default of the first parameter is overwritten, while the second default retains its value.

Now run this program:

```
#include<stdio.h>
void fumbler(int,int);
main()
{
  int boo=10,bap=20;
  fumbler(boo);
}
void fumbler(int foo=0,int beep=0)
{
  printf("%d...%d\n",foo,beep);
}
```

This program will give error. That's because when the function is called and the prototype checked, the default values are not found. Since the compiler first looks at the prototype, the default values also have to be passed in the function prototype definition itself.

So, to a workable program:

```
#include<stdio.h>

void fumbler(int=0,int=0);
main()
{
  int boo=10,bap=20;
  fumbler(boo);
}
void fumbler(int foo,int beep)
{
  printf("%d...%d\n",foo,beep);
}
```

Because the default values have been notified in the function prototype definition, there will be no hurdles to this program. The point we are making is, the definition of the specific function can come later, although its default parameters must be included in the prototype checklist.

The following program won't run again!

```
#include<stdio.h>

void fumbler(int=0,int);

main()
{
  int boo = 10, bap =20;
  fumbler(boo);
}
void fumbler(int foo,int beep)
```

```
printf("%d...%d\n",foo,beep);
```

The error given will be the same as when we didn't pass a single default value.

One would expect the C++ compiler to accept the single parameter passed to it and use the single default value in its prototype to account for the other parameter. But that's not what happens. The default values are overwritten from left to right. Therefore, the default value of foo, which is 0 would be overwritten by the value of boo-10, but since the other default value has not been specified and the second parameter not passed to the function, the error will be generated.

Correct Program:

```
#include<stdio.h>
void fumbler(int,int = 0);
main()
{
  int boo = 10;
  fumbler(boo);
}
 void fumbler(int foo, int beep)
{
  printf("%d...%d\n",foo,beep);
}
```

This time, the value of *boo*, 10, which you have so correctly passed

will be taken by *int foo* and even if you haven't passed the second parameter, the default value in the prototype will have no problems being accepted. *foo* will thus be printed as 10 and *beep* as 0.

Let's keep on at our zigzag routine- interspersing erroneous programs with workable ones. We'll understand better. So, it's time for an erroneous program:

```
#include<stdio.h>

void fumbler(int,int = 0,int);

main()
{
  int boo = 10,bap = 20,zap = 30;
  fumbler(boo);
}
  void fumbler(int foo,int beep,int pop)
{
  printf("%d...%d\n",foo,beep);
}
```

This is again an erroneous program.

What we've got to understand simply, is that the prototype is checked from right to left. Therefore, the default values should also be passed from right to left.

Let's make that program work:

```
#include<stdio.h>
```

```
void fumbler(int,int = 0,int = 0);

main()
{
  int boo = 10;
  fumbler(boo);
}
void fumbler (int foo,int beep,int pop)
{
  printf("%d...%d...%d\n",foo,beep,pop);
}
```

The output you get will be:10...0...0

The default values will be checked from right to left. The compiler will find everything above board and perform flawlessly. The single parameter passed in the function definition will output its prescribed worth. The second parameter will get its value by default, and so will the third parameter.

That is the controlled flexibility of C++. A notch above C, keeping the virility of C intact while adding more structure to that kind of thinking. We'll move on to another aspect of C++. Global scoping. Taking the concept of global variables and extending their reach, or more properly, making them pliable to our wishes. So, jump into the driver's seat and meet the new concept head-on.

Global Scoping

Consider:

```
#include<stdio.h>
```

```
int i = 100;

main()
{
 int i;
 i = 10;
 printf("%d...%d\n",i,::i);
 ::i = 20;
 printf("%d...%d\n",i,::i);
}
```

The output is:

10...100
10..20

Unlike C which makes a global variable inaccessible when a local variable of the same name is available in the block, C++ allows the flexibility for accessing the global variable despite a local presence. With a simple symbol like::prefixing the variable

As the above example would have proved, not only can we access a global variable and print its existing value, we can also reinitialize it to another value. Hence, the latter output:10...20.

Here's another program with a global scope:

```
#include<stdio.h>

int i;

main()
```

```
{
  int i;
  i = 10;
  printf("%d...%d\n",i,::i);
  {
    int i = 20;
    printf("%d...%d\n",i,::i);
  }
printf("%d...%d\n",i,::i);
}
```

i will, be equal to 20 and *::i* will be equal to 0. That's because *::* refers to global scoping. It is not an operator that gives access to variable above the concerned block. When you say, *::i*, therefore, you are referring to the global *i* and not the local *i* one block above.

The output obtained will be:

10...0
20...0
10...0

Referencing

Let's take on another fundamental concept. Referencing. A reference is like a pointer. But, let's tune in to a program before we focus on the concept.

```
#include<stdio.h>

main()
```

```
 int i = 17;
 int &p = i;
 printf("p = %d...i = %d\n",p,i);
 p = 55;
 printf("p = %d...i = %d\n",p,i);
}
```

If we look at the output that this gives us, 17...17 and 55...55, will realize that &p is like pointer. That is exactly what it is. A pointer and more.

A reference, as the name suggests, is like an alias. Call it by this name or that, it "refers" to the same entity. What it the advantage of referencing, then?

Referencing goes a long way in erasing untidiness in code, making it more sophisticated and readable.

Let's take another program and get the essence of references really clear:

```
#include<stdio.h>
main()
{
 int i = 17;

 int &p = i;
 printf("p = %d...i =%d\n",p,i);

 p = 55;
 printf("p = %d...i = %d\n",p,i);
```

```
  i = 90;
  printf("p = %d...i =%d\n",p,i);
}
```

The theme of this program is that a variable and its reference are so tightly inter-locked, that a change in one necessarily results in a change in the other. So, whether we change the value of *i* or *p*, both the values will change concurrently. Cross check that with the output:

```
17...17
55...55
90...90
```

Operators too work on reference variables similarly. For example:

```
#include<stdio.h>
main()
{
  int i = 17;
  int &p = i;
  printf("p = %d...i = %d\n",p,i);
  p++;
  printf("p = %d...i = %d\n",p,i);
i++;
printf("p = %d...i = %d\n",p,i);
}
```

Whether we augment the value of *i* or we augment the value of its reference *p*, both the values will augment similarly. So, look at the output:

```
17...17
```

18...18
19...19

To reiterate that bond further, let's ask for the addresses of the two, the variable *i* and its reference *p*.

```
#include<stdio.h>

main()
{
  int i =17;
  int &p = i;

  printf("%U...%U\n",&i,&p);
}
```

The output is :

65524...65524

Once a reference variable has been defined to refer to a particular variable, it cannot refer to any other variable.

The 'const' Keyword

First, a program:

```
#include<stdio.h>

main()
{
  const int i = 17;
```

```
  int &p = i;
  printf("p = %d...i = %d\n",p,i);

  p = 55;
  printf("p = %d...i =%d\n",p,i);
}
```

The term *const* prefaced to *i* renders the reference powerless to change the value of the variable referred to.

Output:

17...17
55...17

Prefixing or suffixing the *int* is not the issue. Either way, a const is a const is a constant. It rephrases the variable in such a way as to close it to change through a reference.

A const object has to be initialized. That would, of course, be very obvious. If you do not initialize a constant object, what would it be constant at?

A normal variable and a constant variable occupy locations in memory blocks completely removed from each other.

The reference is dependent upon the variable it refers to. So if we change the value of the variable, we change the value of the reference automatically, constant or otherwise.

Just as we were able to manipulate integers through references, it is possible to manipulate strings as well.

Constant Pointers

Constant pointers behave in the same way as constant variables. We cannot modify a constant pointer, just as we couldn't touch a constant variable.

We cannot change the pointer to point to another string, but we can change the elements of the string.

Function Overloading

The idea behind function overloading is comfortingly simple. To appreciate its potential, however, let's first talk about C. In C, two function definitions with the same name in the same scope would generate an error. If you, therefore, wanted a display of an *int*, a *long* and a *string*, you'd need three separate functions to indicate the same function albeit with different datatypes. You'd probably incorporate a *displayi()* for the *int, displayl()* for the long and *displays()* for the *string*. And that would be the simplest collection you'd ever encounter. Think about the number of function names you'd have to remember if you had a fatter collection of datatypes to be displayed.

The C++ compiler, on the other hand, understands the pressures that a programmer has to contend with, and goes about smoothening the path in every way it can. Function overloading is one such road roller. Let's take a program for a practical stint before we get into explanations:

```
#include<stdio.h>

void display(int);
void display (long);
void display(char*);
```

```
main()
{
  int i = 10;
  long j = 600000;
  char *str ="Hello";
  display(i);
  display(i);
  display(str);
}
void display(int var)
{
  printf("%d \n",var);
}
void display(char *var)
{
  printf("%d \n",var);
}
```

display() in the single function name that displays three data types, an *int,* a long and a string. The C++ compiler simply considers the job set aside for the display() function, and decides on which particular display() function should be executed, depending on the parameter(s) passed. The output you'll get when you run this program will be:

10
6000000
Hello

That is, one single display() function to give the three printouts asked for.

Function overloading not only reduces memorization units, but also

does away with lengthy and redundant code.

A point of consideration, however, is that C++ does not distinguish between different return values. For function overloading to work effectively, the differences should be in parameters passed, not in return types.

One thing is must always be remembered while using function overloading. Pass different parameters to distinguish between two functions with the same name. Don't rely on return values to differentiate them.

C++ gives the privilege of declaring variables as and when their need arises. C had a glaring limitation, in that all the variables will ever need in the program, even if it is one instance in the last line of a 200 line program, have to be anticipated and declared right in the beginning. C++ liberates from that straitjacket.
And another trick in the informality circus. The use of *cout* instead of the stiffer *printf(). printf()* demands a certain protocol. We have to state whether we want an *int* to be printed, a *string*, a character, etc. with the appropriate notation. We had to specify a %d, a %s, a%c, a %u, etc. *cout*, which does not require these specifications. it recognizes each datatype for what it's worth, and prints accordingly. Take this program:

```
/*"iostream.h" is the header file
where cout has been defined*/

#include <iostream.h>

main()
{
```

```
        int i = 10;
        cout<<"OOPs we're there";//print a string
        cout<<"i + "<<; // print a string and an int
}
```

The '//' Comment

A trivial, yet noteworthy comment that C++ provides: the use of // instead of the starry /*...*/ to comment out a statement. Take a preceding program as a specimen.

We need the symbol // at that point of our line which we wish to comment out. The part prefixed by that symbol is hidden from the compiler as comment. Line 4 and 5 are examples.

If we wanted to keep an entire block out of the compiler's purview, however, we should still use the C symbol /*...*/.
For single lines the newer // symbol is the better bet.

That covers most of C++ as an extension to C. Small additions and subtractions to the basic C structure, stretching C itself.

C++ is a language that incorporates the concept of object oriented programming or OOPs. It is, in fact, this aspect that gives C++ its aura of innovation. It is this that lifts it out of the C genre and gives it its exclusivity.

Object Oriented Programming

It is Object Oriented Programming, with its newness that gives C++ that ring of novelty. As Scott Robert Ladd, one of the most eloquent preachers of C++ defines it, Object Oriented Programming lets you

organize data in your program the same way as a biologist organizes his sativas and pudicas and different kinds of plants. Yes, Object Oriented Programming has to do with classes and families, genres and species.

Specifically the focus of OOPs is data. Data, that is the soul of any program. After all, we inhabit the world of data processing. In the melee for program modulation and program conciseness, however, we've lost sight of the actual building blocks - the data. We directed all our energies into the body beautiful and let the cells fester. In short, we took data for granted, often forgetting their role, in the larger pursuit.

Enter Object Oriented Programming. Not so much a revolutionary method as a revolutionary attitude. A re-defining of objectives.
OOPs recognizes the import of data as the nucleus of all programming, and attempts to define the relationship between data and operations.

Data is no longer the silent observer, suffering quietly in its shell, while functions walked away with the applause and the credit. Data has finally been lifted out of its innocuous existence and put on par with functions. Object Oriented Programming stresses the dependency of one on the other. Functions cannot exist without data, and data need functions to give it shape.

Objectivity in OOPs

Let's steep ourselves into this radical method of thinking, an attitude which is fast become a cult.

A program to initiate us into the new mould:

```
#include<stdio.h>

struct integer
{
  private:
  int i;
};
main()
{
  struct integer num;
  num.i = 10;
  printf("%d\n",num.i);
}
```

This program will not go beyond the drawing board. All because of the *private: int i* builds a barbed wire around the variable i.

Although i is a member of the structure num, main() cannot invade its private precincts, however correct the rest of the code is. The data thus secured cannot be touched or tampered with, even accidentally.

Let's take another program with a related concept: public. We'll broadcast the explanation after we've seen a program.

```
#include<stdio.h>
struct integer

{
  public:
  int i;
};
main()
{
```

```
  struct integer num;
  num.i = 10;
  printf("%d\n",num.i);
}
```

This program will work without a whimper, because i here is public property.

To summarize the two programs that we breezed through, a private member of the structure is guarded against outside interference, while a public member is for all to use. When we label a member private we literally hide it from the public eye, protecting its substance.

This is the essence of that much touted phrase 'data encapsulation'.

Does that mean that a private member cannot be accessed at all? That it has to necessarily remain rigid and fixed throughout its existence ? Certainly not.

The output will be exactly the same as the previous program. *i* will be initialized through the member function printf() and displayed. Both *i* and *print(void)* are members of the class integer. It is just not possible to give more details of C++ and OOPs here. It is suggested to refer to the books available with you for that purpose.

But before we end this session, it is worthwhile to give a brief introduction to C++ Graphics. For learning and understanding graphics properly, one will have to consult some book.

C++ Graphics

To begin with, when we start with graphics we need a HEADER

FILE called GRAPHICS.H. This contains definitions and explanations of all the functions and constants we will need. So we can consider this our starting point.

Now that we have the definitions of our functions, we can move on to our actual functions. These are kept in the graphics library file, GRAPHICS.LIB.

Both these files are provided as part of TURBO C.

So our minimum requirements to get started with graphics are:
1. the header file GRAPHICS.H
2. the graphics library file GRAPHICS.LIB
3. and a dream.

You will find some files with a BGI extensions and some with a CHR extension. Be sure to transfer them to the directory in which you are currently working.

The Beginning: Switching to the Graphics Mode

We are currently in the test mode, with 25 rows and 80 columns. But now we will switch to graphics by running this program:

```
#include <graphics.h>
void main()
{
 int gm,gd=DETECT;
 printf("BEFORE GOING INTO GRAPHICS MODE\n");
 getch();
 initgraph(&gd,&gm,"");
 printf("AFTER GOING INTO GRAPHICS MODE.AND NOW
QUITTING..\n");
```

```
/*please reboot the machine*/
}
```

This program takes you from the text mode to the graphics mode. What we have done in the program is:

- defined 2 integers *gm* and *gd*, and then
- called the function *initgraph()*

Let us run this program.

The first *printf()* will print in the text mode. The computer then waits for us to press a key. As soon as we press a key, the function *initgraph()* will be called. And then the screen will be cleared, thus reflecting the switch from the text mode to the graphics mode.

Now the second *printf()* will print in the graphics mode and the program will terminate. Now move the cursor to ..... where's the cursor???!!!

Remember, you are in the graphics mode, so the cursor will not be displayed. Graphics modes do not support the conventional cursor.

Now list the directory and see how slowly it scrolls compared to the text mode.

To return to the text mode you will have to reboot the machine. It is good practice to always go back to your starting point, in this case the text mode. This is what we will do in the next program.

```
#include <graphics.h>
void main()
{
```

```
 int gm,gd=DETECT;
 printf("BEFORE GOING INTO GRAPHICS MODE\n");
 getch();
 initgraph(&gd,&gm,"");
 printf("AFTER GOING INTO GRAPHICS MODE.AND NOW
QUITTING..\n");
 getch();
 closegraph();
 printf("AFTER GOING OUT OF GRAPHICS MODE INTO TEXT
MODE\n");
}
```

Again this program will print the first *printf()* in text mode, wait for a key to be pressed, call *initgraph()*, switch to graphics mode, print the next *printf()* in graphics mode and then wait for you to press a key. As soon as you press a key, the function *closegraph()* will be called. It simply closes the graphics mode and restores the screen to its prior mode. Think of it as performing housekeeping chores.

Always use the *closegraph()* function to leave the graphics mode.

Thus, *closegraph()* negates *initgraph()*.

About Graphic Modes and Drivers

We can go to the graphics mode by calling the *initgraph()*. All very straightforward. However, there are various modes within graphics, so we have to decide which graphics mode we require.

To understand this, we need to understand monitors and the concept of resolution. The number of dots or picture elements (pixels) available to us on the screen in the graphics mode, is known as the resolution. The greater the number of dots, the higher the

resolution. Simply put, this means that the more dots available, the clearer our picture.
The function *initgraph()* figures out the resolution and assigns it a number. This number is then put in the address of *gm*. Thus, *gm* is the current graphics mode and specifies the resolution. The *gm* number tells us which monitor we are using, and its resolution, the number of video pages it supports and the colors available.

The variables *gm* and *mode* are equivalent, the only difference being that *gm* is obtained by the function *initgraph()* while *mode* is obtained by the function *getgraphmode()*. The information regarding monitor and resolution, i.e. *gm*, is obtained by calling *getmodename()*. The mode number is passed to this function. And *getmodename()* accordingly returns a pointer to a string, where the required information is stored.

So these are the very basic requirement to start C++ graphics. C++ supports an extensive list of graphics functions like, *bar(), rectangle(), circle()*,.... etc, that makes graphics very easy. It will not be possible to write about or even list these functions here. Kindly refer some book for that purpose. Learning C++ graphics is indeed advisable. You will enjoy it!

CHAPTER 4

Problem Specification

Chapter 4

Problem Specification

4.1 Receptor Oriented Methods

Receptor models which were evolved rapidly over the past few years but have just recently been recognized as a distinct discipline. The general category of receptor models includes both microscopic and chemical methods of apportioning source contributions to ambient air particulates. The number and variations of these methods have grown rapidly over the past few years and include such methods as automated scanning electron microscopy, chemical mass balance and multivariate procedures. These methods as well as hybrid procedures such as target transformation factor analysis, are reviewed and their boundary conditions, strengths, and weaknesses have been discussed by many researchers.

Substantial progress has been made over the past 10 to 15 years in reducing air particulate pollution levels. A large number of regions, however, are still not meeting the National Air Quality Standards despite costly and extensive control efforts. Air pollution control strategies will be more critically evaluated in the future than they have been in the past as industry, government, and the public demand more effective programs in the face of high inflation rates and energy costs.

Development of air particulate control programs by state and federal agencies and evaluation of their effectiveness as new standards and sources that are introduced, require an accurate understanding of the

- chemical and physical characteristics of air particulates,
- their origin, and

- the contribution of each source to air particulate levels.

Although a great deal of the data have been accumulated on the chemical and physical characteristics of air particulates, very little is known about their sources and contributions to specific receptor sites because of the inadequacy of previously available predictive tools.

Source oriented dispersion models based on emission inventories and meteorological parameters have been the primary tools of control agencies to estimate the impact of a particular source at a receptor site. Estimation of source impact on ambient loadings at receptor sites, however, has been approximate at best because many of the variables required by these models, primarily meteorological, are random in nature, vary with space and time and may combine with other variables in a nonlinear manner.

Over the past ten or more years, receptor oriented models have been examined as possible source impact assessment tools. It has only been recently, however, that they have evolved to the point where they now form a distinct discipline and clear alternative to source oriented dispersion models. The number and variations of these receptor models have grown rapidly over the past few years. The objective of this project is to develop a software for some of these receptor models.

Dispersion Models

The urban atmosphere is a highly complicated system composed of gaseous and particulate material. This atmospheric matter is made up of a wide variety of complex organic compounds, inorganic chemicals, and radionuclides with thousands of possible sources. Both natural and anthropogenic sources contribute to this complex mixture

and include meteor dust, sea salt, forest fires, volcanoes, wind blown soil, agricultural tilling, transportation sources, space heating, cooking, industrial processes, etc.

Assessing the impact of individual sources depends on the ability to either predict the impact from fundamental meteorological principles, source oriented models, or the ability to resolve the contributions of this large number of sources after the fact with receptor models as shown in Figure 1. Source oriented dispersion models start with emission rates and dispersion factors for a specific source and calculate the impact at a receptor site. Although source oriented models have the potential to assess the impact of specific sources, none are currently able to describe adequately the complicated random nature of dispersion in the atmosphere and the results in general have been considered only approximate.

Receptor Models

Receptor models, on the other hand, start with the measurement of a specific feature of the aerosol at the receptor and after the fact calculate the contribution of a specific source type. Measurable atmospheric features include particle size, shape and color, particle size distribution(number), component identification (organic, inorganic and radioactive), component chemical state and concentration, time and spatial variations. Although most of these features can be used to identify source types, the only measured parameters that can be used to determine quantitatively a source's contribution to air particulate levels are a component's concentration or the number of particles of a specific type and size.

Receptor models can be conveniently grouped into two basic categories, microscopic and chemical methods, as illustrated in figure 2. Microscopic methods have a high source resolving capability for

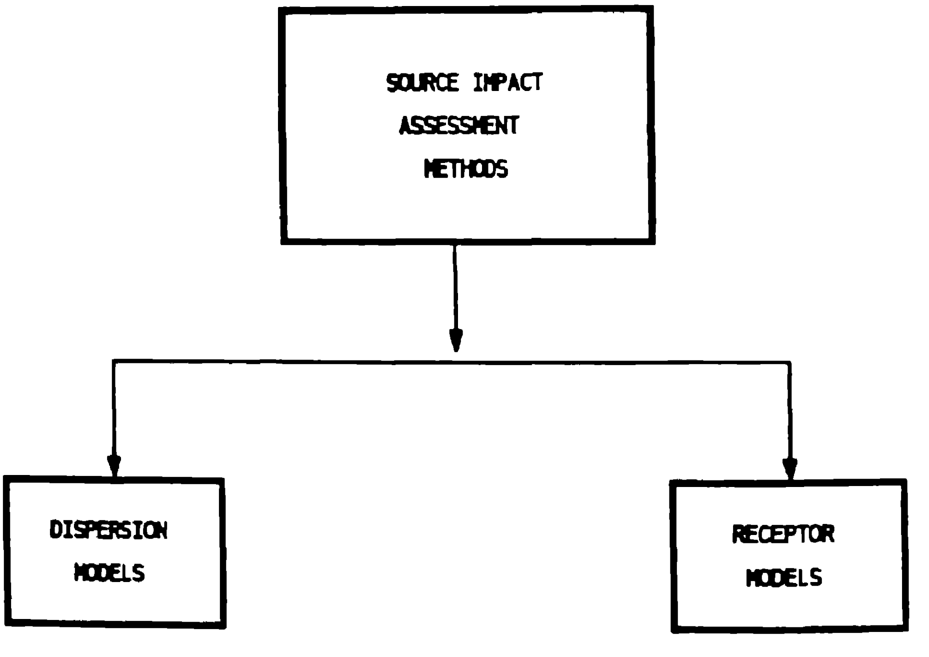

SOURCE IMPACT
ASSESSMENT
METHODS
DISPERSION
MODELS
RECEPTOR
MODELS

FIGURE 1

sources with characteristic morphological features such as wood fibre, tyre rubber, pollen, etc. To be quantitative, however, they must estimate the number of particles, their density and volume, and must analyze enough particles to be representative of the total sample. On the other hand, chemical methods require knowledge of the chemical composition of both the ambient and source particulates and are based on an assumed conservation of mass.

Microscopic Methods

Microscopic methods can be divided into three subcategories, optical, which is limited to particles greater than about 2 m but can use such features as colour to aid in its qualitative identification of particle type; scanning electron microscopic (SEM) which is applicable to smaller particles and may include a determination of the major elements to aid in qualitative particle type assignment; and automated SEM analysis which uses all of the same qualitative particle type identification features as the SEM analysis but have the capability of analyzing more particles because of its automation. This latter method is the latest and promises to extend the general applicability of microscopic methods mainly because it provides for a more representative analysis. It needs to be evaluated further, however, and is still limited by many of the normal problems associated with microscopic methods. These include time and cost per analysis, lack of sensitivity for amorphous organic species which often account for a large fraction of the aerosol and large uncertainties associated with the basic assumptions required for quantitative analysis such as particle density and volume. There is no doubt that microscopic methods have been and always will be useful tools for air particulate source impact assessment but whether or not these methods become widely used for large scale quantitative assessment will depend on the results of their further evaluation, costs and the capabilities of alternate methods.

Chemical Methods

Chemical methods can be divided into five subgroups as shown in figure 2. All of the chemical methods require knowledge of the chemical composition of both the ambient aerosol and possible sources. Enrichment factors, time, and spatial series analyses provide primarily qualitative information about possible sources whose contribution can only be determined quantitatively by applying a special case of the chemical mass balance model, i.e., the tracer method. Chemical mass balance and advanced multivariate data analysis methods are more likely to be widely used in the future for large scale source impact assessment studies.

Multivariate methods such as factor analysis were the first of the chemical receptor models to be used for air particulate source impact assessment. These statistical methods were developed for problems in the social sciences, and applied to the physical sciences in the 1950s before Bliford and Meeker first applied them to air pollution problems in 1967. The following year Prinz and Stratmann also applied factor analysis to air quality problems. Winchester and Nifong analyzed the Chicago area aerosol pattern based on source emission inventories in 1969 and Hidy and Friedlander used a simple tracer form of the chemical mass balance method in 1970 to assess the impact of sources such as automotive exhaust and geological dust using Pb and Si as indicators. Although five papers were published in 1971, it wasn't until 1972 that the formalism of the chemical mass balance approach was firmly established by Miller, Friedlander, and Hidyyand by Friedlander in 1973. Four additional papers appeared in 1972 and ten in 1973. Numerous others have appeared since and within the past year it has become clear that chemical reporter models are coalescing into a clearly defined discipline.

Chemical Mass balance Method

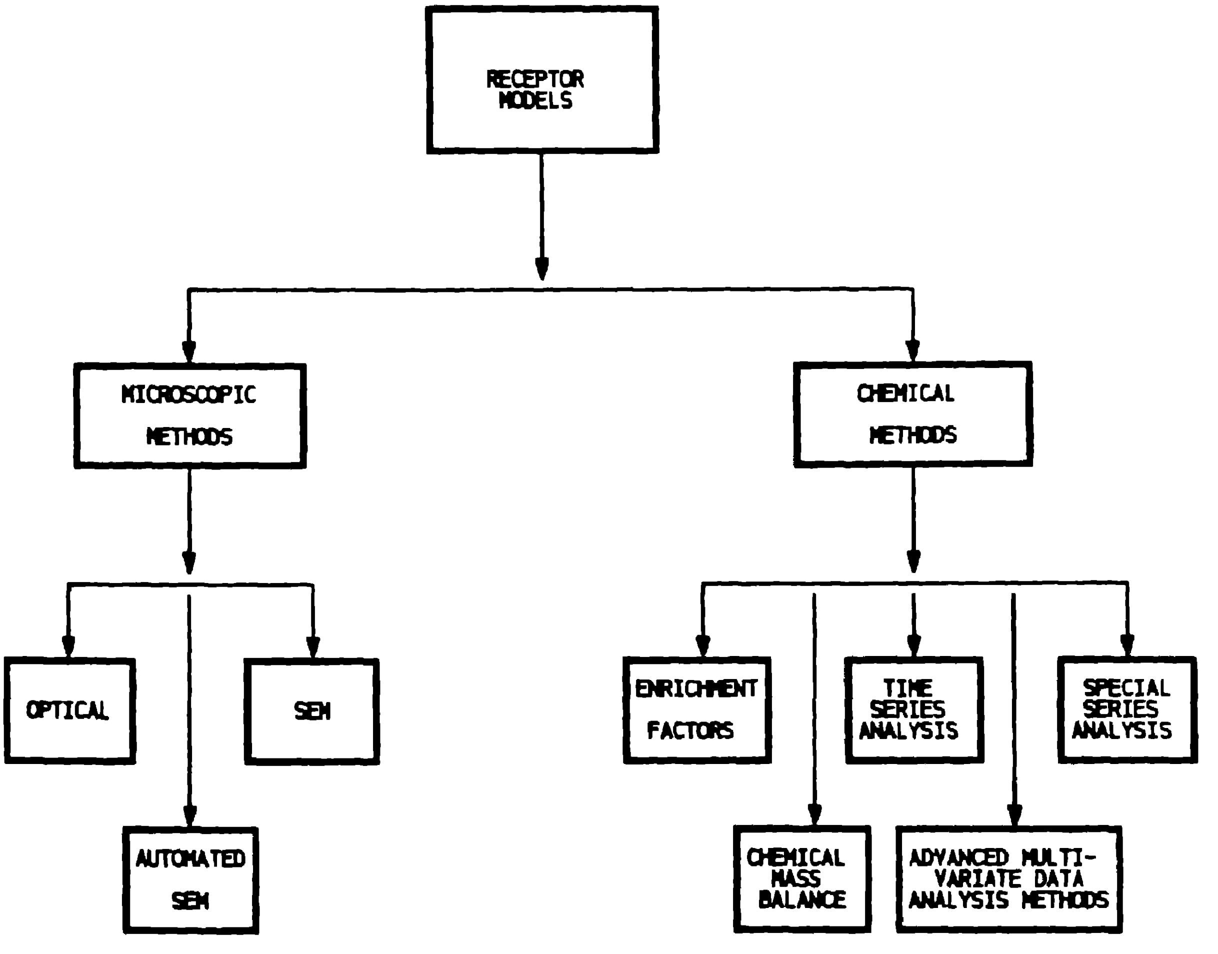
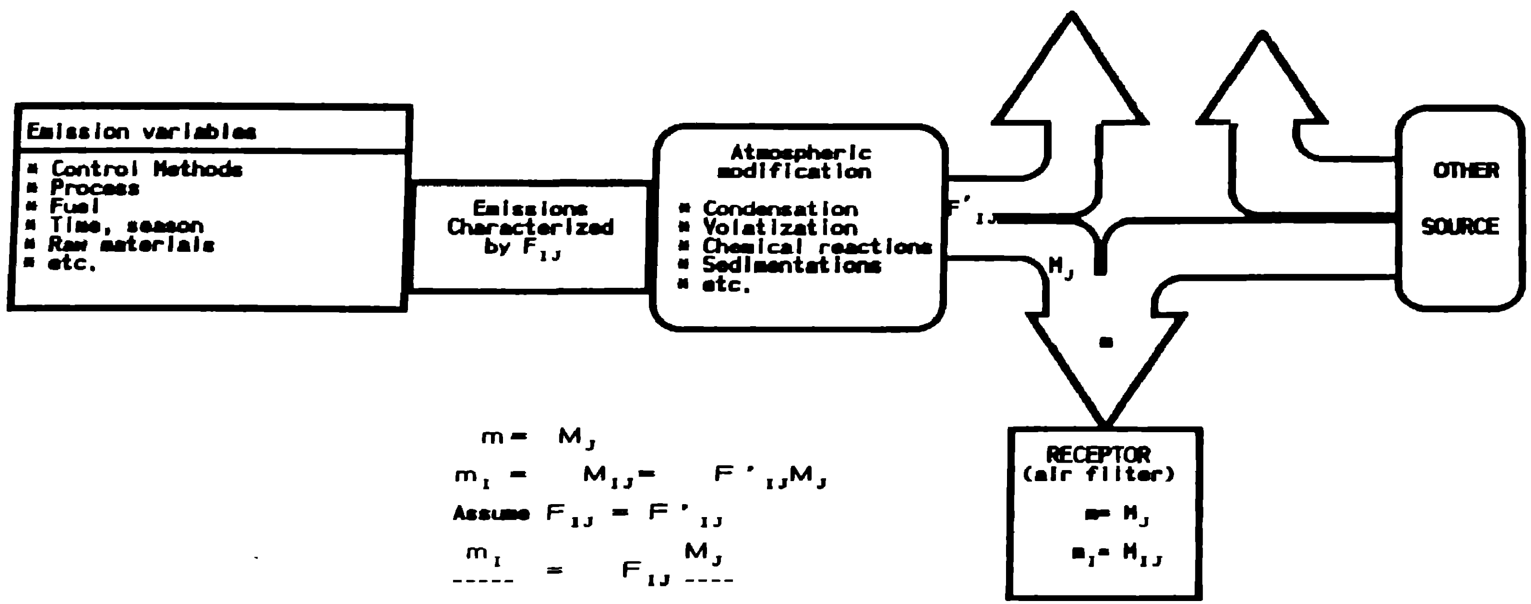

RECEPTOR MODELS
MICROSCOPIC METHODS
CHEMICAL METHODS
OPTICAL
SEM
AUTOMATED SEM
ENRICHMENT FACTORS
TIME SERIES ANALYSIS
SPECIAL SERIES ANALYSIS
CHEMICAL MASS BALANCE
ADVANCED MULTI-VARIATE DATA ANALYSIS METHODS

FIGURE 2.



$m = M_J$

$m_I = M_{IJ} = F'_{IJ} M_J$

Assume $F_{IJ} = F'_{IJ}$

$$\frac{m_I}{m} = F_{IJ} \frac{M_J}{m}$$

$C_I = F_{IJ} S_J$

$100\ S_J$ = Percent contribution of source J

FIGURE 3.

The chemical mass balance (CMB) method is the basis for quantitative source impact determinations and is discussed first and in the greatest detail. It is based on the principles established by Miller, Friedlander, and Hidy and Friedlander. Friedlander's CMB method of source type identification has been applied in Pasadena, Chicago, Fresno, Pomona, San Jose, Riverside, New York, Portland, Washington, D.C. and St. Louis. These applications of the CMB methods suffered, however, major shortcomings related in part to the fact that the data sets to which they were applied were not developed with the needs of the CMB method in mind. These previous applications also had the following shortcomings:

- All major sources were not characterized at the time of the study.
- Source uncertainties were not taken into account.
- The methods used had not been verified nor their sensitivities to variations in input parameters established.
- Ambient and source aerosol characterization was incomplete i.e., some of the major chemical species such as carbon, SO_4, NO_3, Si, etc. were not determined.

The basic principles of the CMB method as first presented by Friedlander are illustrated schematically in Figure 3. The CMB methods assumes that aerosol mass is conserved from the time a chemical species is emitted from its source to the time it is measured at a receptor. That is, if p sources are emitting M_j mass of particulates,

$$m=\sum_{j=1}^{p} M_j \tag{1}$$

where m is the total mass of the particulate collected on a filter at a receptor site.

The mass of a specific chemical species, m_i, is given by the following

$$m_i=\sum_{j=1}^{p} M_{ij}=\sum_{j=1}^{p} F_{ij}' M_{ij} \tag{2}$$

Where M_{ij} is the mass of element if from source j and F'^{ij} is the fraction of chemical species i in the mass from source j collected at the receptor. It is assumed that the mass of each individual chemical component is conserved so that

$$F_{ij}'=F_{ij} \tag{3}$$

where F_{ij} is the fraction of chemical i emitted by source j as measured at the source. (This same assumption is required by dispersion models and most receptor models by varying degrees.) The degree of validity in this assumption depends on the chemical and physical properties of the species and its potential for atmospheric modifications such as condensation, volatilization, chemical reactions, sedimentations, etc.

Accepting this assumption and dividing both sides of Eq. (1) by the total mass of the deposit collected at the receptor site, it follows that

$$C_i=\sum_{j=1}^{p} F_{ij}S_j \tag{4}$$

where C_i is the concentration of the chemical component i measured at the receptor (air filter) and S_j is the source contribution; i.e., the ratio of the mass contributed from source j to the total mass collected at the receptor site. In practice, it is this fraction of particulate pollution measured at the receptor due to source j, S_j, which is of primary interest in CMB calculations.

If the C_i and the F_{ij} at the receptor for all p of the source types suspected of affecting the receptor are known, and $p \leq n$ (n = number of chemical species), a set of n simultaneous equations exist from which the source type contribution S_j may be calculated by least squares methods.

4.2 The Mathematical Models

The development of receptor-based models has progressed significantly in the past few years. Mathematical receptor models presently in use can be classified into several categories: Chemical mass balance (CMB) multivariate, microscopic and source-receptor hybrids. Here we discuss CMB and multivariate models. Several new models of both types had been presented. First, receptor models are distinguished from conventional dispersion modeling. Next, various approaches to solving the CMB equation for source contributions are given.

Dispersion models are the conventional means of predicting the environmental impact of an emission source on air quality. In general, the dispersion model states that the contribution of source j to a receptor, S_j , is the product of an emissions rate, E_j , and a dispersion factor, D_j , so that

$$S_j = D_j E_j \qquad (1)$$

Dispersion formulae are usually classified by the geometric form of the emission source, giving rise to expressions for point, line, and area sources. A basic aim of air quality meteorology is to relate the dispersion factor for a given source geometry to known meteorological parameters such as wind speed and direction and atmospheric

stability. Current development of dispersion models focuses on modeling dispersion in rough terrain, long range transport, wet and dry acidic deposition, and complex atmospheric chemistry.

In contrast to dispersion models, receptor models start with observed ambient airborne particle concentrations at a receptor and seek to apportion the observed concentrations between several source types based on knowledge of the compositions of the source and receptor materials.

The foundation of all receptor based models for particle source assignment is a simple mass conservation argument. If a number of sources, p, exists, and if there is no interaction between their aerosols that causes mass removal or formation, the total airborne particulate mass measured at the receptor, C, will be a linear sum of the contributions of the individual sources S_j

$$C=\sum_{j=1}^{p} S_j \tag{2}$$

Similarly, the mass concentration of aerosol property i, will be

$$C_i=\sum_{j=1}^{p} a_{ij}S_j \tag{3}$$

where a_j is the mass fraction of source contribution j possessing property i at the receptor.

Chemical Mass Balance Models

When the property i is a chemical property, equation (3) represents a chemical mass balance. If one measures n chemical properties of both source and receptor, n equations of the form of equation (3)

exist. If the number of source types contributing those properties is less than or equal to the number of equations, that is, if p ≤ n, then the source contributions S_j can be calculated by soling the over determined system of liner Equations (3). Five methods of performing this calculation have been applied: the tracer property, liner programming, ordinary linear least-squares fitting, effective variance least-squares fitting,and ridge regression. Each of these methods is discussed below.

Tracer Element Method

The tracer element method is the simplest. It assumes that each aerosol source type possesses a unique property that is common to no other source type. Equation (3) then reduces to

$$S_j = \frac{C_t}{a_{tj}} \qquad (4)$$

for each source *j* with its own tracer *t* (using the same notation as in Equation [3]). It works well when the tracers meet the following requirements: (1) a_{tj} perceived at the receptor is well known and invariant between source and receptor, (2) C_t , can be measured accurately and precisely in the ambient sample and (3) the concentration of property *t* at the receptor comes only from source type *j*.

These conditions cannot be completely met in practice, and limiting the model to only one tracer property per source type means that valuable information contained in the other aerosol properties is discarded. However, in some instances, one can use a tracer approach as a first-approximation, upper limit estimate of source impact. Solutions to the set of Equations (3) have been developed to make use

of the additional information provided by more than one chemical property of a source type, and even by other properties (such as particle shape and size). The other methods of solution must deal with the case of a number of constraints, n, greater than the number of unknowns, p.

Linear Programming Method

Henry had applied a linear programming algorithm that maximizes the sum of the source contributions subject to the following constraints

$$0 \le S_j \le M, \quad and \tag{5}$$

$$C_i + 3\sigma_{C_i} \ge \sum_{j=1}^{p} a_{ij} S_j \tag{6}$$

where M is the total particulate mass concentration and σ_{C_i} is the uncertainty in the measurement C_i. However, this solution has not been developed further.

Ordinary Linear Least-Squares Method

In the ordinary linear least-squares method, the most probable values of S_j when $n > p$ are achieved by minimizing X^2, the sum of the squares of the differences between the measured values of the C_i measurement

$$\chi^2=\sum_{i=1}^{n}\frac{\left(C_i-\sum_{j=1}^{p}a_{ij}S_j\right)^2}{\sigma^2_{C_i}} \qquad (7)$$

Written in matrix terms, the CMB equation (equation [3]) is

$$C=AS \qquad (8)$$

Where C is the n x 1 vector of observed concentrations, A is the n x p source composition matrix, and S is the p x 1 vector of source contributions. The ordinary weighted least-squares solution to Equation (8) is

$$S=(A^tWA)^{-1}A^tWC \qquad (9)$$

where W is a diagonal matrix with $\sigma_{C_i}^{-2}$ on the diagonal. Superscript "t" denotes matrix transpose and "-1" is the matrix inverse. This solution provides the added benefit of being able to propagate the measurement uncertainty of C_i through the calculations in order to develop a confidence interval around the calculated S_j. The error estimates of source contributions calculated from ordinary weighted least-squares, assuming errors only in the observed concentrations and no errors in the source composition matrix, are given by the diagonal elements of the matrix

$$(A^{t}WA)^{-1} \tag{10}$$

Effective Variance Least-Squares Method

The ordinary weighted least-squares solution is incomplete because the ambient aerosol chemical properties C_i are not the only measured observables. The source aerosol chemical properties a_{ij} are also measured observables, but the errors associated with those measurements are not included in the ordinary weighted least-squares fit.

Dunker (1979) and Watson (1979) have applied a treatment to the solution of a set of Equations (3). The maximum likelihood solution minimizes the function

$$\chi^2=\sum_{i=1}^{n}\frac{\left(C_i-\sum_{j=1}^{p}a_{ij}S_j\right)^2}{\sigma^2_{C_i}+\sum_{j=1}^{p}\sigma^2_{a_{ij}}S^2_j} \tag{11}$$

where $\sigma_{a_{ij}}$ is the uncertainty associated with the a_{ij} measurement.

The least-squares solution using uncertainties in the A matrix is identical in form to the ordinary weighted least-squares solution in Equation (9) except that the weights $\sigma_{a_{ij}}$ are replaced by

$$V^{-1}_{eff,i}=\left(\sigma^2_{C_i}+\sum_{j=1}^{p}\sigma^2_{a_{ij}}S^2_j\right)^{-1} \tag{12}$$

where V_{eff} is the effective variance.

Since the effective variance solution depends on the source contributions, which are unknown, an iterative procedure is followed. At each iteration, the previously estimated S_j are used to compute new effective variance weights which are used, in turn, to compute new S_j. The process terminates when the S_j do not change much from step to step. The error estimates for the S_j are obtained from the diagonal elements of the matrix

$$(A^t V^{-1}_{eff} A)^{-1} \qquad (13)$$

where V^{1}_{eff} is a diagonal matrix with terms from Equation (12) on the diagonal.

This solution provides two benefits. First, it propagates a confidence interval around the calculated S_j which reflects the cumulative uncertainty of all the input observables. The more precise the ambient and source property concentration measurements are, the better the source contribution estimate will be. The second benefit provided by this effective variance weighing is to give those chemical properties with larger measurement uncertainties, or chemical properties that are not as unique to a source type, less weight in the fitting procedure than those properties that are more precisely measured or that are truly unique to a source.

Ridge Regression

The CMB Equation (8) can be discussed in statistical terminology as a linear regression problem. In that framework, the observed concentrations C are the dependent variables, the columns of the source composition matrix A are the values of the independent variable, and the source contributions are the regression coefficients to be estimated by least-squares techniques. Ridge regression uses this statistical terminology.

CHAPTER 5

The Mathematical Concepts

Chapter 5

The Mathematical Concepts

In this chapter it has been intented to give some basic concepts of mathamatics that are very much lifeline of the models for which this application is developed. Although the Elimination Method is essentially as it is represented here, the Least-Squares Method involves mostly the matrix manupulation, in the model.

5.1 System of Linear Equations

We are familiar with the problem of finding the values of unknowns, x_1, x_2, etc., which satisfy systems of equations of the type :

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots + a_{3n}x_n = b_3 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n \end{array} \quad (1)$$

We know that for systems of this type, when the number of equations is equal to the number of unknowns, there will ordinarily be a unique solution, i.e., one set of values of $x_1, x_2, \ldots ,x_n$ which satisfy all of the equations. We find that the methods of finding this solution which are learned in a college algebra course are not necessarily the best ones for computer use, and we also find that problems of this type which are so large as to be out of the question for hand calculations are easily done on the computer.

5.1.1 The Elimination Method

The elimination method is used in elementary algebra for solving systems of linear equations. In this method, an appropriate multiple of the first equation is added to each of the other equations so that the resulting *n-1* equations have zero coefficients for the x_1 term. (If the first equation does not have a term involving x_1, we must first interchange two equations to obtain one with an x_1 term as the first equation.) Then an appropriate multiple of the next equation is added to each of the remaining equations so that the x_2 term is eliminated from these. (Again, if the second equation contains no x_2 term, we would first have to interchange two equations.) The process is continued until a system of the following form is obtained:

$$\begin{aligned} b_{11}x_1 + b_{12}x_2 + b_{13}x_3 + \ldots + b_{1n}x_n &= d_1 \\ b_{22}x_2 + b_{23}x_3 + \ldots + b_{2n}x_n &= d_2 \\ b_{33}x_3 + \ldots + b_{3n}x_n &= d_3 \\ &\vdots \\ b_{nn}x_n &= d_n \end{aligned} \qquad (2)$$

Then from the last equation we have $x_n = d_n / b_{nn}$, and by substituting this result in the next to last equation we find x_{n-1}, and so forth, until x_1 is finally found from the first equation.

5.2 Curve-fitting

One of the fields of elementary mathematics where the digital computer has most to contribute is that of curve-fitting. This process is one which is extremely laborious for hand computation, so much so that only quite simple curve-fitting operations were ordinarily done before the day of the digital computer. With the digital computer, however, standard types of programs are available that allow very rapid determination of the best-fitting curve from a wide family of possible choices.

The basic problem in curve-fitting can be described as follows. We have a set of measured values of a quantity x, and an associated set of measured values of another quantity y. We wish to find some functional relation:

$$y = f(x) \tag{3}$$

between x and y which is satisfied by our sets of measured values (x_1, y_1), (x_2, y_2), etc. and which will allow us to infer reasonable values of y for other values of x, where we have no measurement. What do we mean by "reasonable" values of y? Suppose we are having five points (x_1, y_1)......(x_5, y_5), and three different curves passing through all five points. All pass through all the given points. How are we to decide which of these is the best fit? If we have no further information upon which to act, we are intuitively drawn to the smooth curve as the most reasonable fit. This is the type of curve-fitting generally used for interpolation. Quite frequently, however, we know that for physical reasons the functional relation $y = f(x)$ must have some particular function form, e.g.:

$$y = c_1 e^{c_2 x} \tag{4}$$

or perhaps:

$$y = c_1 \tan x + c_2 \cos c_3 x \tag{5}$$

where the constants c_1, c_2, c_3 are not known. In such cases, in attempting to find the best fit we are restricted choosing the constants to give the most reasonable fit. We cannot even be sure in this case that our "best" function will pass through all the given points. Ordinarily it will be impossible to choose the constants so that this will happen. The usual practical problem in curve-fitting is of

the following type. We have some set of values (x_1, y_1), (x_2, x_2),....,(x_n, y_n), which we wish to fit with a curve of the form:

$$y = f(x, c_1, c_2, c_3, \ldots, c_m) \tag{6}$$

where the function f is completely specified except for the particular values of the constants $c_1, c_2, \ldots, c_m$. We wish to choose these constants so that our measured values will most nearly satisfy the above expression.

Again, what do we mean by "most nearly satisfy?" Once we have selected the constants $c_1, \ldots, c_m$, if we substitute in our measured values we will find that the equation:

$$f(x_1, c_1, c_2, c_3, \ldots, c_m) - y_1 = 0 \tag{7}$$

is not satisfied exactly, but instead, there is some error in each case:

$$\begin{aligned} f(x_1, c_1, c_2, \ldots, c_m) - y_1 &= \delta_1 \\ f(x_2, c_1, c_2, \ldots, c_m) - y_2 &= \delta_2 \\ &\vdots \\ f(x_n, c_1, c_2, \ldots, c_m) - y_n &= \delta_n \end{aligned} \tag{8}$$

the quantities $\delta_1, \delta_2, \ldots, \delta_n$ representing the amount of error in the fit. Depending upon the type of fit desired, we might define that fit to be best that causes the quantity $|\delta_1| + |\delta_2| + \ldots + |\delta_n|$ to be a minimum, and accept as our solution the values of $c_1, c_2, \ldots, c_m$ which cause this to happen. As another example, we might define that fit to be best that causes the quantity $\delta_1^2 + \delta_2^2 + \ldots + \delta_n^2$ to be

minimum, and accept as our solution the values of c_1, c_2,....,c_m which cause this to happen. Ordinarily, the latter set of c's will turn out to be different from the preceding set. This latter criterion of goodness of fit, the so called "least squares" criterion, is by far most widely used.

5.2.1 Linear formulae

An extremely wide variety of curve-fitting problems can be grouped into one category, and can be solved with what is basically one machine program. These consist of all curve fitting problems where the undetermined constants appear only in linear form, that is to say, the functional relation between x and y is of the form:

$$y = c_1 f_1(x) + c_2 f_2(x) + \ldots + c_m f_m(x) \qquad (9)$$

where the functions $f_1(x)$, $f_2(x)$, $f_3(x)$, etc. are completely known functions of x. In this case if we have n sets of measured values, (x_1, y_1), (x_2, x_2),...,(x_n, y_n), and we substitute these into the given equation, we would like to satisfy the equations :

$$\begin{aligned} c_1 f_1(x_1) + c_2 f_2(x_1) + \ldots + c_m f_m(x_1) &= y_1 \\ c_1 f_1(x_2) + c_2 f_2(x_2) + \ldots + c_m f_m(x_2) &= y_2 \\ &\vdots \\ c_1 f_1(x_n) + c_2 f_2(x_n) + \ldots + c_m f_m(x_n) &= y_n \end{aligned} \qquad (10)$$

Since the quantities x_1, x_2 etc. are known numbers, the quantities $f_1(x_1)$, $f_2(x_1)$, etc.are known numbers, so the above equations are just a set of n linear equations for the m unknowns, c_1, c_2,....,c_m. If n is equal to m, there is exactly one solution (provided that the determinant of coefficients is not zero), i.e., there is one set of values for c_1, c_2,....,c_m that will cause the above equations to be satisfied. If n is less than m, there are less equations than unknowns and there

are an infinite number of solutions, each of which represents a curve passing through the given points. The problem in this case is not completely determined, and we cannot find a single "best" fit without further information of some sort. If n is greater than m, there are more equations than unknowns, and there is no solution. In this case we look for a solution in the least squares sense. We write the equations in the form:

$$\begin{aligned} c_1 f_1(x_1) + c_2 f_2(x_1) + \dots + c_m f_m(x_1) - y_1 &= \delta_1 \\ c_1 f_1(x_2) + c_2 f_2(x_2) + \dots + c_m f_m(x_2) - y_2 &= \delta_2 \\ &\vdots \\ c_1 f_1(x_n) + c_2 f_2(x_n) + \dots + c_m f_m(x_n) - y_n &= \delta_n \end{aligned} \tag{11}$$

The quantities δ_1, δ_2, etc. are the residuals. For a perfect fit these residuals would all be zero. Since we cannot make them be zero, we try instead to make them all small by looking for the values of $c_1, c_2, \dots, c_m$ which will minimise the expression $\delta_1^2 + \delta_2^2 + \dots + \delta_n^2$. In curve-fitting, this last situation, in which there are more equations than unknowns, is the common one.

5.2.2 The Least Squares Fit

The equations (11) can be abbreviated:

$$\sum_{i=1}^{m} c_i f_i(x_j) - y_j = \delta_j, \quad j = 1, 2, \dots, n \tag{12}$$

and we desire to choose the c_i's so that:

$$\sum_{j=1}^{n} \delta_j^2 = \underline{minimum} \tag{13}$$

This last quantity is to be regarded as a function of the c_i's. A necessary condition for it to be a minimum is that its partial derivatives by each of the c_i's must vanish, i.e.:

$$\frac{\partial}{\partial c_k}\sum_{j=1}^{n}\delta_j^2 = 0 \qquad for\ k = 1,2,\ \dots\ ,m$$

$$or \tag{14}$$

$$\sum_{j=1}^{n}\delta_j\frac{\partial\delta_j}{\partial c_k} = 0 \qquad for\ k = 1,2,\ \dots\ ,m$$

From (12):

$$\frac{\partial\delta_j}{\delta c_k} = f_k(x_j) \tag{15}$$

Substituting this, and the value for δ_j from (12) into (14), we have:

$$\sum_{j=1}^{n}[\sum_{i=1}^{m} c_i f_i(x_j) - y_j]\, f_k(x_j) = 0 \qquad for\ k = 1,2,\ \dots\ ,m$$

$$or$$

$$\sum_{i=1}^{m} c_i \sum_{j=1}^{n} f_i(x_j)\, f_k(x_j) = \sum_{j=1}^{n} y_j f_k(x_j) \qquad for\ k = 1,2,\ \dots\ ,m \tag{16}$$

This is a system of m linear equations for the m unknowns $c_1,c_2,\dots$-$,c_m$, and can be solved.

5.3 The Multivariate Techniques

Multivariate analysis can be characterized as all statistical methods which simultaneously analyze more than two variables on a sample of observations. Thus these methods are extensions of univariate analysis, and bivariate analysis (including correlation and simple regression used to analyze two variables). It is based on three

judgements the researcher must make about the nature and utilization of his data: (1) Are some of the variables dependent on others, thereby requiring special treatment? (2) If yes, how many are to be treated as dependent in a single analysis? and (3) What are the presumed properties of data? The technique to be utilized will depend upon the answers to these three questions.

5.3.1 Factor Analysis

The birth of factor analysis is generally ascribed to Charles Spearman. His monumental work in developing a psychological theory involving a single general factor and a number of specific factors goes back to 1904 when his paper "General Intelligence, Objectively Determined and Measured " was published in the American Journal of Psychology. Of course, his 1904 investigation was only the beginning of his work in developing the Two-Factor Theory, and his early work is not explicitly in terms of "factors". Perhaps a more crucial article, certainly insofar as the statistical aspects are concerned, is the 1901 paper by Karl Pearson in which he sets forth "the method of principal axes". Nevertheless, Spearman, who devoted the remaining forty years of his life to the development of factor analysis, is regarded as the father of the subject.

A statistical study involves a group of individuals with some common attributes. The term "individual" is used here in a generic sense to stand for such objects or entities as persons, census tracts, businesses, etc. Measurements made on such individuals, or attributes of these entities, ar designated simply as variables. Throughout this text the letter N is used for the total number of individuals, and the letter n for the number of variables. A particular variable is denoted by X_j, or simply j, which may be any one of the n variables:1,2,3...,n. The index i is employed to designate any one of the N individuals:1,2,3,...N. Then, the value of a variable X_j for individual i is represented by X_{ji}, the other of the subscripts being of importance. A particular X_{ji} is called an observed value which is measured from an arbitrary origin and by an arbitrary unit.

Some simple statistical concepts are presented here for ready

reference. Frequent use is made of the sum of the N values for a variable X_j, namely:

$$\sum_{i=1}^{N} X_{ji} \text{ or } \sum X_{ji} \tag{17}$$

where, in the second expression, the summation is understood to be over all values of the variable. Furthermore, the index i will be reserved explicitly to refer to the individuals, that is, for the range 1 to N.

For a sample of N observations, the *mean* of any variable is defined by

$$\overline{X_j} = \sum X_{ji}/N \tag{18}$$

The *n* population means would be designated $_j$ if they were required in any theoretical development. The observed values of the variables may be transformed to more convenient form by fixing the origin and the unit of measurement. When the origin is placed at the sample mean, a particular value

$$x_{ji} = X_{ji} - \overline{X_j} \tag{19}$$

is called a *deviate*

The *sample* variance of variable X_j is defined by

$$s^2{}_j = \sum x^2{}_{ji}/N \tag{20}$$

The population variances are denoted by σ^2_j. Now, taking the samples standard deviation as the unit of measurement, the *standardized value* of *j* for individual *i* is given by

$$z_{ji} = x_{ji}/s_j \tag{21}$$

The set of all values z_{ji} (i = 1,2,....,N) is called a variable z_j in

standard form. Obviously the variance of z_j is unity.

For any two variables j and k and sample *covariance* is defined by

$$s_{jk} = \sum x_{ji} x_{ki} / N \tag{22}$$

and the corresponding population parameter is designated σ_{jk}. The *correlation coefficient* in the universe is ρ_{jk} and in the sample is defined by

$$r_{jk} = s_{jk} / s_j s_k = \sum z_{ji} z_{ki} / N = \sum x_{ji} x_{ki} / \sqrt{\sum x^2_{ji} \sum x^2_{ki}} \tag{23}$$

The correlations among all the variables of a study are usually computed as the initial step in a factor analysis.

It is the object of factor analysis to represent a variable z_j in terms of several underlying *factors*, or hypothetical constructs. The simplest mathematical model for describing a variable in terms of several others is a linear one, and that is the form of representation employed here. However, there are still several alternatives within the linear framework, depending on the objective of the analysis. A distinction between two objectives can be made immediately, namely: (1) to extract the maximum variance; and (2) to "best" reproduce the observed correlations.

An empirical method for the reduction of a large body of data so that a maximum of the variance is extracted and fully developed as the method of *principal components*, or *component analysis*. The model for component analysis is simply:

$$z_j = a_{j1}F_1 + \ldots + a_{jn}F_n \qquad (j = 1, 2, \ldots, n) \tag{24}$$

where each of the n observed variables is described linearly in terms of n new uncorrelated components $F_1, F_2, \ldots, F_n$. An important property of this method, insofar as the summarization of data is concerned, is

that each component, in turn, makes a maximum contribution to the sum of the variances of the n variables. For a practical problem only a few components may be retained, especially if they account for a large percentage of the total variance. However, all the components are required to reproduce the correlations among the variables.

In contrast to the maximum variance approach, the *classical factor analysis* model is designed to maximally reproduce the correlations. The basic factor analysis mode may be put in the form:

$$z_j = a_{j1}F_1 + a_{j2}F_2 + \ldots + a_{jm}F_m + d_jU_j \qquad (j = 1, 2, \ldots, n), \tag{25}$$

where each of the n observed variables is described linearly in terms of m (usually much smaller than n) *common factors* and a *unique factor*. The common factors account for the correlations among the variables, while each unique factor accounts for the remaining variance (including error) of that variable. The coefficients of the factors are frequently referred to as "loadings".

Principal Factor Method

Principal factor method involves the diagonalization of the matrix **R** by performing a sequence of orthogonal transformations on it, designed to reduce *one* off-diagonal element to zero at each stage.Here **R** is the reduced correlation matrix having communalities at the diagonal. To find communalities, the inverse of the original correlation matrix is obtained say R'. Then the communalities are solved using the formula

$$\mathbf{h}_j = 1\text{-}1/\mathbf{r'}_{jj}$$

where r'_{jj} is the diagonal element of R'. This method of finding the communalities is called squared. Multiple correlation (SMC).

Each orthogonal transformation is of the form $\mathbf{B}_{jk}\mathbf{R}\mathbf{B}'_{jk}$ where

$$
B_{jk} = \begin{bmatrix} 1 & & & & & & \\ & \cdot & & & & & \\ & & \cdot & & & & \\ & & & \cos\theta_{jk} & -\sin\theta_{jk} & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & \sin\theta_{jk} & \cos\theta_{jk} & & \\ & & & & & \cdot & \\ & & & & & & \cdot \\ & & & & & & 1 \end{bmatrix} \qquad (26)
$$

with the four elements in the intersections of the rows and columns j and k as indicated, all other diagonal elements unity, and all other elements zero. The angle of rotation θ_{jk} is chosen so as to transform the element r_{jk} into zero, and is given by

$$
\tan 2\theta_{jk} = \frac{-2r_{jk}}{h^2_j - h^2_k} \qquad (27)
$$

When an element is reduced to zero it does not, in general, remain at zero during subsequent transformations. However, the sum of squares of off-diagonal elements is decreased each time by an amount corresponding to $2r^2_{jk}$, and thus Jacobi's method guarantees the convergence of the off-diagonal elements of **R** to zeros (to a designated number of decimal places) with a sufficient number of iterations.

In the process of reducing the original matrix **R** to the diagonal matrix **D**, the off-diagonal elements are considered systematically. Since the original correlations r_{jk} are altered by the transformations (26), it would seem more appropriate to designate a general off-diagonal element by d_{jk} and the intermediate derived matrices by **D**'s with suitable subscripts indicated below. Thus, at any stage of a specific transformation directed at reducing one off-diagonal element d_{jk} to zero, the following notation will be employed:

$$
{}_{v}D = B_{jk(v-1)}DB_{jk} \qquad (28)
$$

in which $j = 1,2,\ldots,n-1$ and $k = j+1, j+2, \ldots, n$, producing the $n(n-1)/2$ possible pairs (j,k).

At any stage the product of the ν individual transformations may be designated

$$ {}_{\nu}B = B_{jk}\ldots B_{13}B_{12}, \qquad (29) $$

and when *all* combinations of j and k have been tried, the product of these transformations in the i^{th} iteration is designated:

$$ B_i = \prod (B_{jk};\ \ j < k = 1,2,\ldots,n), \qquad (i = 1,2,\ldots,\mu) \qquad (30) $$

The diagonalizing effect of the i^{th} iteration is given by:

$$ D_i = B_i D_{i-1} B'_i \qquad (i = 1,2,\ldots,\mu) \qquad (31) $$

If the product (in the indicated order) of the transformation matrices through the i^{th} iteration is defined by:

$$ P_i = B_i B_{i-1}\ldots B_2\ B_1, \qquad (32) $$

then the formula (31) can be expressed in terms of the original matrix **R** as follows:

$$ D_i = P_i\ RP'_i \qquad (33) $$

After a specified number of iterations , the solution to the problem will be in the form:

$$ D_{\mu} = B_{\mu} D_{\mu-1} B'_{\mu} = BRB' = D \qquad (34) $$

where

$$B = P_{\mu} = B_{\mu}B_{\mu-1}\ldots B_2B_1. \quad (35)$$

The diagonal elements λ_p of **D** are the eigenvalues of **R**, and the rows of the final transformation matrix **B** contain the corresponding eigenvectors

$$\alpha_p = (\alpha_{1p}, \alpha_{2p}, \ldots, \alpha_{np}) \quad (36)$$

of **R**. Since each individual transformation matrix is orthogonal (and hence the product of such matrices is orthogonal), the resulting final matrix **B** is orthogonal and therefore the eigenvectors are normalized. then the coefficients of each factor F_p are obtained simply by multiplying the square root of the eigenvalue by its associated eigenvector, namely:

$$a_p = \alpha_p\sqrt{\lambda_p}. \quad (37)$$

In general, when the rank m of **R** is equal to its order n, there will be n real non negative eigenvalues, and coefficients for n common factors will ensue. If, however, the reduced correlation matrix **R** is positive semi-definite and of rank $m < n$ then there will be $n - m$ zero eigenvalues and there will be only m common factors with non zero coefficients. Of course, if the Gramian properties are not satisfied by the reduced correlation matrix, then some of the eigenvalues will be negative and the coefficients of the corresponding factors will be imaginary.

Quartimax Method

After finding the factors by principal factor method, a transformation is desired which will tend to increase the large factors loadings and decrease the small ones for each variable of the original factor matrix. In this attempt to increase the inequalities among the factors loadings, the size implied is independent of algebraic sign. The inequalities among squares of factor loadings are to be maximized; or, more specifically, that the variance in the distribution of squared factor loadings should be made a maximum by the use of

orthogonal transformations. This approach involves the maximization of fourth powers of factors loadings.

The object of the quartimax method is to determine the orthogonal transformation **T** which will carry the original factor matrix **A** into a new factor matrix **B** for which the variance of squared factor loadings is a maximum.

For any variable z_j, the orthogonal transformation in the plane of factors p and q through an angle ϕ will carry the original coordinates (a's) into the new co-ordinates (b's) as follows:

$$\begin{aligned} b_{jp} &= a_{jp}\cos\phi + a_{jq}\sin\phi \\ b_{jq} &= -a_{jp}\sin\phi + a_{jq}\cos\phi \end{aligned} \qquad (38)$$

According to the basic equations of transformation in a plane. In order to measure the effect of such a transformation $\mathbf{T}_{pq}$ on the overall criterion Q, the sum of fourth powers of the new loadings of the two rotated factors is determined. This sum is defined by

$$Q_{pq}(\phi) = \sum_{j=1}^{n} (b^4{}_{jp} + b^4{}_{jq}), \qquad (39)$$

which depends on the parameter Φ. The object is to determine the angle of rotation (Φ_{pq} in precise notation) for any pair of factor p and q which will make the sum Q_{pq} a maximum. Then the product of the transformations of all combinations of pairs of factors produces a transformed factor matrix:

$$B = AT_{12}T_{13}\ldots T_{pq}\ldots T_{(m-1),m} \qquad (40)$$

where $p = 1,2,\ldots,(m-1)$, and the associated $q = p + 1, p + 2,\ldots,m$. The complete set of $m(m-1)/2$ pairings of p and q is called a *cycle*. In each *cycle* of operations the value of Q for the entire matrix is as large as

or larger than the preceding sum. Since the theoretical maximum of fourth powers of factor loadings cannot exceed n (even when the total unit variance of each variable is analyzed), the procedure must coverage after a sufficient number of cycles.

For any rotation T_{pq} the angle Φ which will make Q_{pq} a maximum can be determined using:

$$\tan 4\phi = \frac{2\sum_{j=1}^{n} (2a_{jp}a_{jq})(a^2_{jp} - a^2_{jq})}{\sum_{j=1}^{n}\left[(a^2_{jp} - a^2_{jq})^2 - (2a_{jp}a_{jq})^2\right]} \qquad (41)$$

Varimax Method

The varimax method a modification of the quartimax method which more nearly approximates simple structure.
The varimax rotation requires that the final factor loadings be such as to maximize the function:

$$V = n\sum_{p=1}^{m} \sum_{j=1}^{n} (b_{jp}/h_j)^4 - \sum_{p=1}^{m}\left(\sum_{j=1}^{n} b^2_{jp}/h^2_j\right)^2 \qquad (42)$$

The computing procedure for a varimax solution is quite similar to that employed for a quartimax solution, but requires that (42) be maximized. Factors are rotated two at a time according to the scheme indicated in (40), and the complete cycle of $m(m-1)/2$ pairings of factors is repeated until the value of V to the specified number of decimal places no longer increases.

It will be convenient to introduce some additional notation in order to make the subsequent expressions more compact. First, the "normalized" factor loadings of a variable z_j for a particular pair of factors p and q, will be designated by

$$x_j = a_{jp}/h_j \qquad (43)$$
$$y_j = a_{jq}/h_j$$

and the rotated loadings by X_j, Y_j, so that the transformation may be put in the form:

$$(X_j \quad Y_j) = (x_j \quad y_j)\begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \qquad (44)$$

where ϕ is the angle of rotation in the plane of the factors p and q. Since squares and cross-products of the normalized loadings are required in the computation, the following notation will be employed:

$$\begin{aligned} u_j &= x^2_j - y^2_j \\ v_j &= 2x_j y_j \\ A &= \sum u_j \\ B &= \sum v_j \\ C &= \sum (u^2_j - v^2_j) \\ D &= 2\sum u_j v_j \end{aligned} \qquad (45)$$

where all sums are on j from 1 to n

The required angle of rotation is

$$\tan 4\phi = \frac{D - 2AB/n}{C - (A^2 - B^2)/n} \qquad (46)$$

CHAPTER 6

The Logic

Chapter 6

The Logic

It is very logical to get into the very logic of anything that you want to learn and master so it will be illogical if I don't provide even with the superficial logic of the programs here. We shall consider them one by one. But before that I must inform you that all the menu*.c programs are developed using the C++ **graphics**

Menu1.c:

This program doesn't do anything but informs the user that this software is for 'Air Quality Modelling'. But still it uses a lot of graphics capabilities. A low-level animation is also attempted here.

Menu2.c:

This program provides the user with three option.

1. Data   2. Analysis   3. Exit

the option 1) is meant for different types of data and data file management. On selecting this option the program control goes to the program menu3.c

the option 2) provides with different methods of data manipulation. On selecting this option the control is sent to the program menu4.c.

the option 3) is for leaving the software

Menu3.c

data management. The option are:

1. Create a data file.

   On selecting this option, we are provided with three options further based on which the data files are given extensions of:
   i. New line ELM, if the file is to be used in elimination method.

   ii. LSQ, if the file is to be used in least squares method.

   iii. FAC, if the file is to be used for Factor Analysis.

   Depending upon the option, the file is created.

2. Edit a data file.

   Again we have three options under this option for specifying the type of the data file as mentioned above.

3. Copy a file.

   Here the name of the file that is to be copied is asked first and then the name of the file where it is to be copied is read. Then making use of the *system()* statement, The file is copied.

4. Rename a file:

   The name of the file to be renamed is read as the first string and the new name is read as the second string. Then making use of *strcat()* and *system()* statements the file is renamed.

5. Delete a file:

   The selected file is deleted following the procedure similar to above.

6. List of files:

The prompt appears to input the path of the list (dir) to be shown. Then again we make use of *system()* statement.

7. On selecting this the current program is exit.

Menu4.c:

The control comes to this program if we select the option 2. (Analysis) in the program Menu2.c

Here we are supposed to choose from the three options;

1. Chemical Mass Balance:

- If we select this option, the control goes to the program Menu5.c where the input source (Screen File) and output destination (Screen, Printer or File) is selected and then control passes to menu6.c.

2. Factor Analysis:

On selecting this option, the control goes to the program Menu5.c where the input source and output destinations are selected and then control passes to menu7.c.

3. This option is for leaving this menu. The control goes back to menu2.c. on selecting this option.

Menu5.c

This program is used for selecting one out of two options for input source:

1. Screen
2. File

It again prompts to opt for the output destination out of three choices:

1. Screen
2. Printer
3. File.

Depending on these choices, a value between 1 through 6 is passed back by this program to the program which activated it. But before asking to enter the choice, the user is asked whether he wants to continue. If the answer is no, control goes back to menu4.c.

Menu6.c:

If we choose for Chemical Mass Balance option in menu4.c, the control comes to this program. Here we have options for :

1. The Elimination Method:

   Depending on the value returned by menu5.c, an adequate program amongst elm1.c,elm2.c,...,elm6.c is executed.

2. The Least Square Method

   Depending on the value returned by menu5.c, an adequate program amongst lsq1.c,lsq2.c,...,lsq6.c is executed.

3. Exit this menu:

   On selecting this option the control is returned back to menu5c..

Menu7.c:

This program is executed if we choose for Factor Analysis option in menu4.c. In this program a menu is generated again

which provides with the following three choices:

1. Rotation 1

   Following this option, the initial factor loadings are obtained by Principal-Factor Method and then the Quartimax Rotation is applied to it.

2. Rotation 2

   This option provides with the Varimax Rotation after solving for initial factor loadings using Principal Factor Method.

3. Exit this menu:

   As usual! for exiting the current menu!

All the above mentioned programs, with the word 'menu' in their names are mostly meant for creating menus. Most of them involve initiating graphics mode and graphics commands. As it might be clear by now, you require the number keys for most of the time to run this software other than while in putting a name. The remaining menus which we get are mostly in the textmode only. Let us proceed with some of the calculative programs. This software involves the matrix manipulation extensively. So far avoiding the excessive coding, general function are written for matrix-transpose, multiplication, inverse, correlation, etc. which are called in other programs as and when need arises. I shall discuss them briefly here.

Transpose.c :

The function accepts a two-dimensional array, stores the transpose of it in another array and passes the new array as a parameter. The logic for this is very simple. The rows of the original matrix are converted into columns (or vice-versa) and stores into the new array.

Mat-mul.c:

This function expects seven parameters: first matrix, and its dimensions, second matrix and its dimensions, the resultant matrix. At first a test is made to check if the number of columns of first matrix is equal to the number of rows of the second matrix. If not, error is triggered otherwise the general logic of matrix multiplication is incorporated to get the resultant into the last parameter. The logic is very elementary which we study in the school. So I don't feel a need to write the details here.

Mat-inv.c:

The square matrix, its one dimension (row or column) and the resultant matrix are transferred as the arguments to this function. An identity matrix is augmented to the initial matrix. The row and column operations are performed on this augmented matrix so that, as a result, finally we get a matrix in which first appears the identity matrix and augmented to it, the inverse of the original matrix.

Correl.c:

Here we pass the original matrix, its dimensions and the resultant matrix as the parameters. In this function correlation among the different variables is calculated and the resultant correlation matrix is passes back.

Factor.c:

This function gets in the parameters, the correlation matrix and its one dimension (since # rows = columns), an array containing the eigenvalues. After getting the correlation matrix, the method of chapter 5 is applied and the eigenvalues, factor loadings and variances are passes back. For logic refer (5.3.1).

Qrtmax.c:

We pass the array of initial factor loadings found by factor.c, its dimensions and the resultant loadings' matrix as arguments to this function. Following the procedure of Quartimax Rotation, as described in Chapter 5, the initial factor loadings are rotated and the resultant rotated factor loadings are passed back through the last parameter.

Varimax.c:

This function accepts the array of initial factor loadings, found by factor.c, its dimensions, array of communalities and the resultant loadings' array as parameters. Varimax rotation is applied to the initial factor loadings and the resultant rotated factor loadings are passed back through the last parameter.

Elm*.c:

All the programs elm1.c, elm2.c,...,elm6.c are similar logicwise except the input source and output destination variations. These programs solve for Elimination Method. The procedure has already been described (5.1.1).

lsq*.c:

As in the case of elm*.c the only difference between the programs lsq1.c, lsq2.c,..., lsq6.c is that of the input source and output destination. Kindly refer to (4.2) and (5.2.2) for understanding the procedure. These programs are meant for Least-Squares Method.

qmax*.c:

The differences are again similar as above between qmax1.c... qmax6.c. See (5.3.1). These programs find the rotated factor loadings using Quartimax Rotation.

Varimax*.c:

Same differences. Procedure already explained in (5.3.1). Meant for Varimax Rotation.

Same differences. Procedure already explained in (5.3.1). Meant for Varimax Rotation.

In all the above files referred by ___*.c, the pattern is like this:

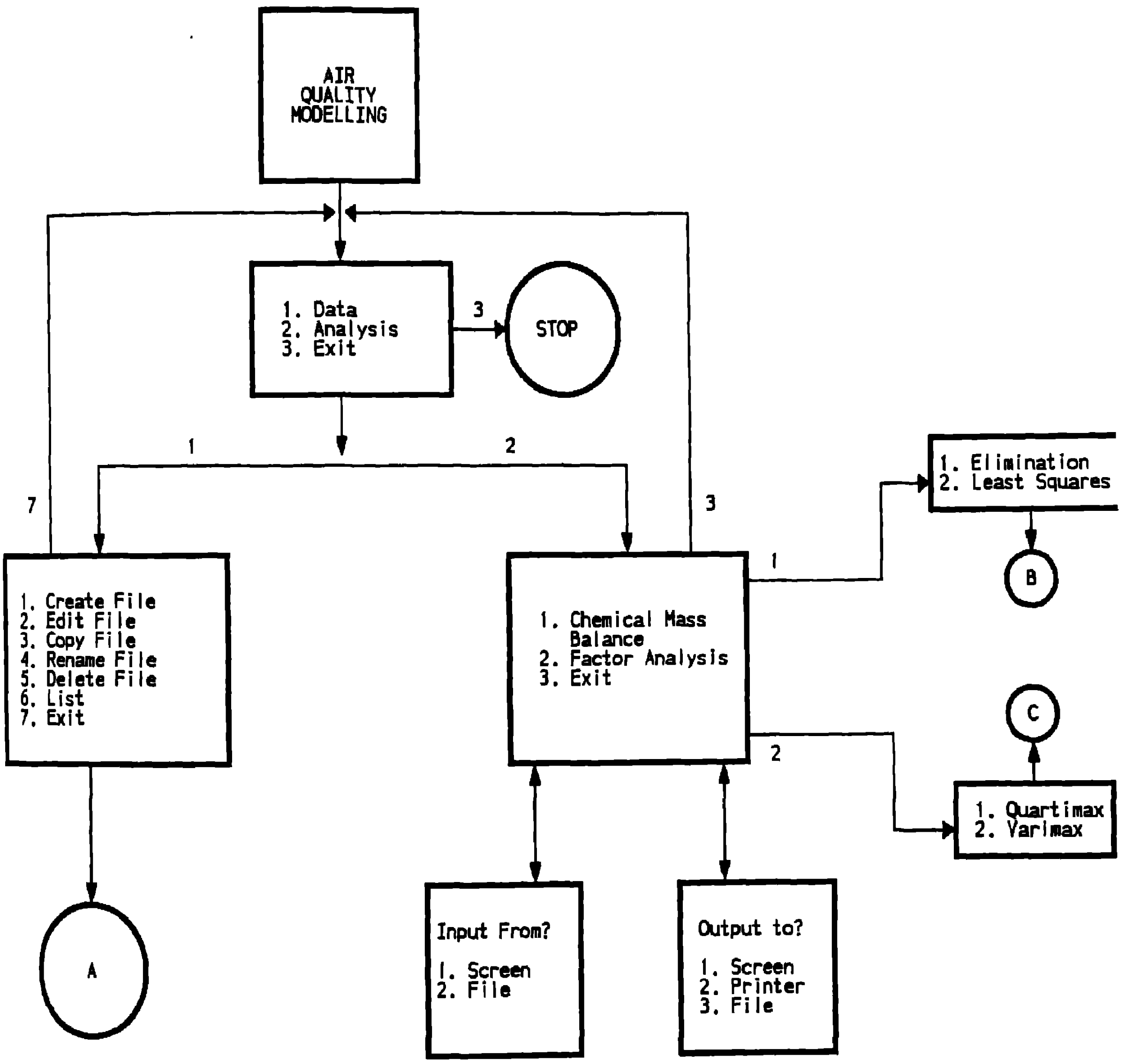
	Input sources	Output destination
1.	Screen	Screen
2.	-do-	Printer
3.	-do-	File
4.	File	Screen
5.	-do-	Printer
6.	-do-	File

Main.c:

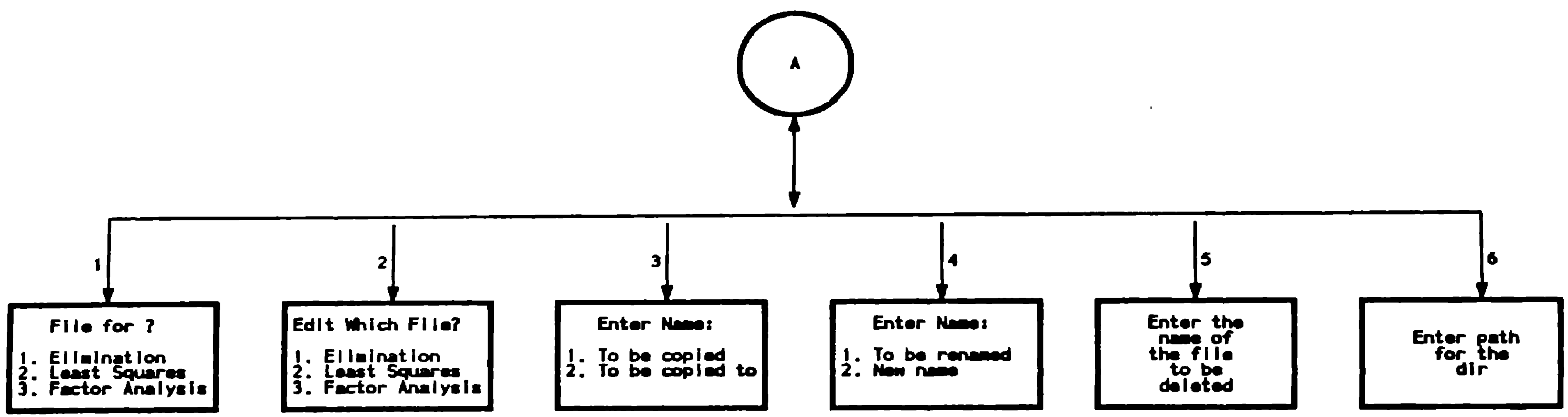
This program controls the execution of all the programs depending upon the different options provided by the user.

CHAPTER 7

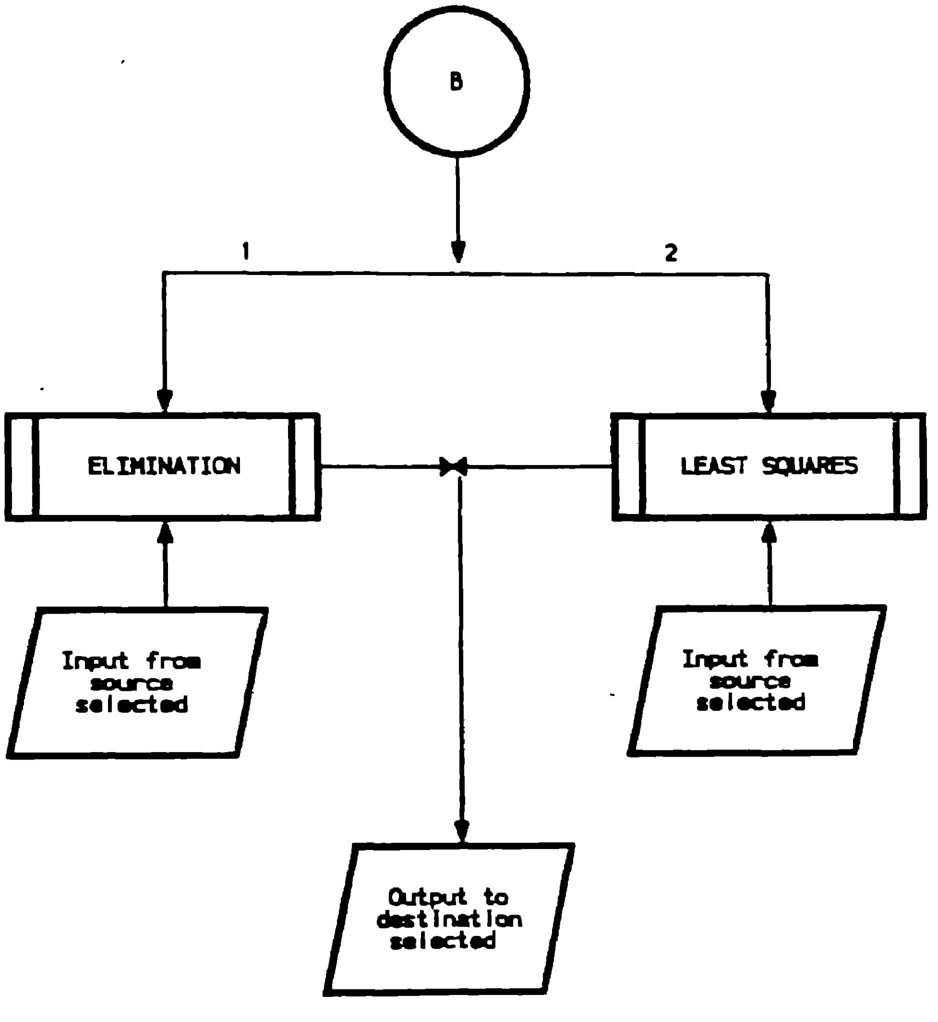
The System's Flowchart

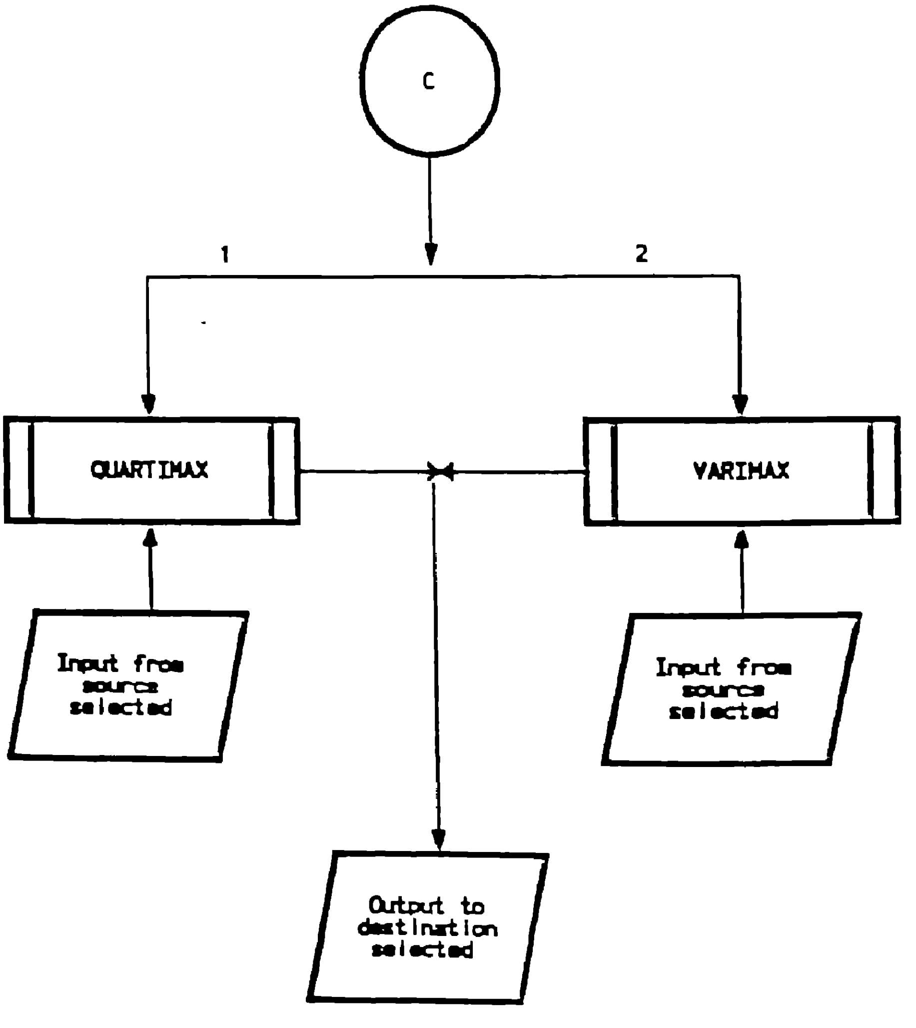


THE SYSTEM'S FLOWCHART



A
1
2
3
4
5
6
File for ?
1. Elimination
2. Least Squares
3. Factor Analysis
Edit Which File?
1. Elimination
2. Least Squares
3. Factor Analysis
Enter Name:
1. To be copied
2. To be copied to
Enter Name:
1. To be renamed
2. New name
Enter the name of the file to be deleted
Enter path for the dir


B
1
2
ELIMINATION
LEAST SQUARES
Input from source selected
Input from source selected
Output to destination selected


C
1
2
QUARTIMAX
VARIMAX
Input from source selected
Input from source selected
Output to destination selected

APPENDIX A

Programs Listing

```
/* CORRELATION */

#include <math.h>
#include <stdio.h>
#define  MAX  50
void correl(float x[MAX][MAX],int nsp,int nr,float r[MAX][MAX])
{
  int i,j,il;
  float sum,suml,sum2,sum3,temp[50][50],xmean[50];


  for (j=0;j<nsp;j++)
  {
    sum = 0;
    for (i=0;i<nr;i++)
      sum = sum + x[i][j];
    xmean[j] = sum/nr;
  }

  for (j=0;j<nsp;j++)
    for (i=0;i<nr;i++)
      temp[i][j] = x[i][j] - xmean[j];


  for (i=0;i<nsp-1;i++)
    for (j=i+1;j<nsp;j++)
    {
      suml = sum2 = sum3 = 0;
      for (il=0;il<nr;il++)
      {
              suml = suml + (temp[il][i] * temp[il][j]);
              sum2 = sum2 + (temp[il][i] * temp[il][i]);
              sum3 = sum3 + (temp[il][j] * temp[il][j]);
      }
      r[i][j] = suml/sqrt(sum2*sum3);
    }
  for (i=0;i<nsp;i++)
    r[i][i] = 1.0;
  for (i=0;i<nsp-1;i++)
    for (j=i+1;j<nsp;j++)
      r[j][i] = r[i][j];
}
```

```
/*********ELM1.C*************/

/***********************************/
/********MATRIX-ELIMINATION*******/
/*********SCRN - SCRN*************/
/***********************************/


#include <stdio.h>
#include <stdlib.h>


void main()
{
int n,i,j,l,k,max_row,k1,k2;
float a[30][30],ans[30],max,temp,pivot,mul;
char  name[30],*name1[30],*name2[30];


clrscr();
gotoxy(10,5);
highvideo();
cprintf("\tM A T R I X   E L I M I N A T I O N\n\n");
normvideo();
gotoxy(5,10);
printf("Kindly give the #source (= #specious) ");
scanf("%d",&n);
flushall();
gotoxy(5,10);
clreol();
printf("Name of the next source ? ");
for (i=0;i<n;i++)
 {
  gotoxy(31,10);
  clreol();
  name1[i]=malloc(30);
  *name1[i]=NULL;
  gets(name);
  strcpy(name1[i],name);
 }
gotoxy(5,23);
highvideo();
cprintf("PRESS ANY KEY...!!!");
normvideo();
getch();
flushall();
{
```

```
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
gotoxy(5,10);
printf("Name of the next species ? ");
for (i=0;i<n;i++)
{
  gotoxy(32,10);
  clreol();
  name2[i]=malloc(30);
  *name2[i]=NULL;
  gets(name);
  strcpy(name2[i],name);
}
gotoxy(5,23);
highvideo();
cprintf("PRESS ANY KEY...!!!");
normvideo();
getch();
flushall();
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
gotoxy(10,12);
highvideo();
cprintf("Please input the coefficients : ");
normvideo();
for (i=0;i<n;i++)
{
   for (j=0;j<=n;j++)
   {
   gotoxy(42,12);
   clreol();
   scanf("%f",&a[i][j]);
   }
}
gotoxy(5,23);
highvideo();
cprintf("PRESS ANY KEY...!!!");
normvideo();
getch();
fflush(stdin);
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
gotoxy(10,10);
printf("The given matrix is :\n");
for (i=0;i<n;i++)
{
  for (j=0;j<=n;j++)
    printf("%f  ",a[i][j]);
```

```
  printf("\n");
|
gotoxy(5,23);
highvideo();
cprintf("PRESS ANY KEY...!!!");
normvideo();
getch();
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
flushall();
for (i=0;i<n-1;i++)
{
  max = abs(a[i][i]);
  max_row = i;
  for (j=i;j<n;j++)
  {
    if(abs(a[j][i]) > max)
    {
      max = a[j][i];
      max_row = j;
      for (k=0;k<=n;k++)
      {
            temp = a[i][k];
            a[i][k] = a[max_row][k];
            a[max_row][k] = temp;
      }
    }
  }
    pivot = a[i][i];
    if (pivot == 0.0)
    {
      for (i=6;i<25;i++)
      {
            gotoxy(1,i);
            clreol();
      }
  err : gotoxy(10,10);
        printf("********* Solution doesn't exist ***********\n");
        gotoxy(5,23);
        highvideo();
        cprintf("PRESS ANY KEY...!!!");
        normvideo();
        getch();
        flushall();
        for (i=6;i<25;i++)
        {
              gotoxy(1,i);
              clreol();
        }
        flushall();
        exit(0) ;
      }
      for (l=i+1;l<n;l++)
      {
        mul = a[l][i];
        for (j=i;j<=n;j++)
              a[l][j] = a[l][j] - ((a[i][j] * mul)/pivot);
      }
    }
    if (a[n-1][n-1] == 0.0)
```

```
    goto err;
  ans[n-1] = a[n-1][n]/a[n-1][n-1];
  for (i=n-2;i>=0;i--)
  {
    temp = 0;
    for (j=n-1;j>=i+1;j--)
    temp=temp+(a[i][j]*ans[j])/a[i][i];
    ans[i]=(a[i][n]/a[i][i])-temp;
  }
  gotoxy(10,10);
  printf("The source strengths are found to be :\n\n");
  flushall();
  for (i=0;i<n;i++)
    printf("%s\t:  %f\n",name1[i],ans[i]);
  gotoxy(5,23);
  highvideo();
  cprintf("PRESS ANY KEY...!!!");
  normvideo();
  getch();
  flushall();
  clrscr();
}
```

```
/********ELM2.C************/


/**********************************/
/*********MATRIX-ELIMINATION******/
/**********SCRN - PRN*************/
/*********************************/



#include <stdio.h>
#include <stdlib.h>


void main()
{
int n,i,j,l,k,max_row,k1,k2;
float a[30][30],ans[30],max,temp,pivot,mul;
char  name[30],*name1[30],*name2[30];



 clrscr();
 gotoxy(10,5);
 highvideo();
 cprintf("\tM A T R I X   E L I M I N A T I O N\n\n");
 normvideo();
 gotoxy(5,10);
 printf("Kindly give the #source (= #specious) ");
 scanf("%d",&n);
 gotoxy(5,23);
 highvideo();
 cprintf("PRESS ANY KEY...!!!");
 normvideo();
 getch();
 flushall();
 for (i=6;i<25;i++)
 {
   gotoxy(1,i);
   clreol();
 }
 gotoxy(5,10);
 printf("Name of the next source ? ");
 for (i=0;i<n;i++)
 {
   gotoxy(31,10);
   clreol();
   name1[i]=malloc(30);
   *name1[i]=NULL;
   gets(name);
   strcpy(name1[i],name);
 }
 gotoxy(5,23);
 highvideo();
 cprintf("PRESS ANY KEY...!!!");
 normvideo();
 getch();
 flushall();
 for (i=6;i<25;i++)
 {
   gotoxy(1,i);
```

```
  clreol();
}
gotoxy(5,10);
printf("Name of the next species ? ");
for (i=0;i<n;i++)
{
  gotoxy(32,10);
  clreol();
  name2[i]=malloc(30);
  *name2[i]=NULL;
  gets(name);
  strcpy(name2[i],name);
}
gotoxy(5,23);
highvideo();
cprintf("PRESS ANY KEY...!!!");
normvideo();
getch();
flushall();
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
gotoxy(10,12);
highvideo();
cprintf("Please input the coefficients : ");
normvideo();
for (i=0;i<n;i++)
{
   for (j=0;j<=n;j++)
   {
   gotoxy(42,12);
   clreol();
   scanf("%f",&a[i][j]);
   }
}
gotoxy(5,23);
highvideo();
cprintf("PRESS ANY KEY...!!!");
normvideo();
getch();
flushall();
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
fprintf(stdprn,"\nThe given matrix is :\n\n");
fprintf(stdprn,"Species : \t");
for (i=0;i<n;i++)
  fprintf(stdprn,"%s\t\t",name2[i]);
fprintf(stdprn,"\n\n\n");
for (i=0;i<n;i++)
{
  fprintf(stdprn,"%s\t",name1[i]);
  for (j=0;j<=n;j++)
    fprintf(stdprn,"%f\t",a[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");
flushall();
for (i=0;i<n-1;i++)
```

```
{
  max = abs(a[i][i]);
  max_row = i;
  for (j=i;j<n;j++)
  {
    if(abs(a[j][i]) > max)
    {
      max = a[j][i];
      max_row = j;
      for (k=0;k<=n;k++)
      {
              temp = a[i][k];
              a[i][k] = a[max_row][k];
              a[max_row][k] = temp;
      }
    }
  }
    pivot = a[i][i];
    if (pivot == 0.0)
    {
err : fprintf(stdprn,"********* Solution doesn't exist ***********\n");
      flushall();
      exit(0) ;
    }
    for (l=i+1;l<n;l++)
    {
      mul = a[l][i];
      for (j=i;j<=n;j++)
              a[l][j] = a[l][j] - ((a[i][j] * mul)/pivot);
    }
}
   if (a[n-1][n-1] == 0.0)
      goto err;
   ans[n-1] = a[n-1][n]/a[n-1][n-1];
   for (i=n-2;i>=0;i--)
   {
     temp = 0;
     for (j=n-1;j>=i+1;j--)
     temp=temp+(a[i][j]*ans[j])/a[i][i];
     ans[i]=(a[i][n]/a[i][i])-temp;
   }
   highvideo();
   fprintf(stdprn,"\nThe source strengths are found to be :\n\n");
   normvideo();
   flushall();
   for (i=0;i<n;i++)
     fprintf(stdprn,"%s\t:  %f\n",name1[i],ans[i]);
   clrscr();
  }
```

```
/********ELM3.C************/


/*********************************/
/********MATRIX-ELIMINATION*******/
/*********SCRN - FILE*************/
/*********************************/



#include <stdio.h>
#include <stdlib.h>



void main()
{
int n,i,j,l,k,max_row,k1,k2;
float a[30][30],ans[30],max,temp,pivot,mul;
char  name[30],*name1[30],*name3[30],c;
FILE  *fp;



clrscr();
gotoxy(10,5);
highvideo();
cprintf("\tM A T R I X    E L I M I N A T I O N\n\n");
normvideo();
a  :  gotoxy(5,7);
printf("Kindly give the name of the output file :  ");
gets(name);
fp = fopen(name,"r");
if (fp)
{
  gotoxy(5,15);
  highvideo();
  cprintf("FILE ALREADY EXISTS ...!!!...");
  gotoxy(5,17);
  cprintf("\tOVERWRITE (Y/N) ....????");
  normvideo();
b : c=getch();
  switch(c)
  {
    case 'y' :
    case 'Y' : break;
    case 'n' :
    case 'N' : {
                                    gotoxy(5,17);
                                    clreol();
                                    gotoxy(5,15);
                                    clreol();
                                    fcloseall();
                                    flushall();
                                    gotoxy(48,7);
                                    clreol();
                                    goto a;
                                }
    default         : {
                                    sound(100); delay(100);
                                    sound(200); delay(100);
```

```
                                        nosound();
                                        goto b;

        }
        fclose(fp);
        gotoxy(5,17);
        clreol();
        gotoxy(5,15);
        clreol();
        gotoxy(48,7);
        clreol();
        fp = fopen(name,"w");
        if (!fp)
        {
          gotoxy(5,15);
          highvideo();
          cprintf("Can't open the file ...!!!...");
          delay(1500);
          gotoxy(5,15);
          clreol();
          normvideo();
          gotoxy(48,7);
          clreol();
          goto a;
        }
        gotoxy(5,10);
        printf("Kindly give the #source (= #specious) ");
        scanf("%d",&n);
        gotoxy(5,23);
        highvideo();
        cprintf("PRESS ANY KEY...!!!");
        normvideo();
        getch();
        flushall();
        for (i=6;i<25;i++)
        {
          gotoxy(1,i);
          clreol();
        }
        gotoxy(5,10);
        printf("Name of the next source ? ");
        for (i=0;i<n;i++)
        {
          gotoxy(31,10);
          clreol();
          name1[i]=malloc(30);
          *name1[i]=NULL;
          gets(name);
          strcpy(name1[i],name);
        }
        gotoxy(5,23);
        highvideo();
        cprintf("PRESS ANY KEY...!!!");
        normvideo();
        getch();
        flushall();
        for (i=6;i<25;i++)
        {
          gotoxy(1,i);
          clreol();
        }
        gotoxy(5,10);
```

```
printf("Name of the next species ? ");
for (i=0;i<n;i++)
{
  gotoxy(32,10);
  clreol();
  name2[i]=malloc(30);
  *name2[i]=NULL;
  gets(name);
  strcpy(name2[i],name);
}
gotoxy(5,23);
highvideo();
cprintf("PRESS ANY KEY...!!!");
normvideo();
getch();
flushall();
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
gotoxy(10,12);
highvideo();
cprintf("Please input the coefficients : ");
normvideo();
for (i=0;i<n;i++)
{
   for (j=0;j<=n;j++)
   {
   gotoxy(42,12);
   clreol();
   scanf("%f",&a[i][j]);
   }
}
fprintf(fp,"\nThe given matrix is :\n");
fprintf(fp,"Species : \t");
for (i=0;i<n;i++)
  fprintf(fp,"%s\t\t",name2[i]);
fprintf(fp,"\n\n");
for (i=0;i<n;i++)
{
  fprintf(fp,"%s\t",name1[i]);
  for (j=0;j<=n;j++)
    fprintf(fp,"%f\t",a[i][j]);
  fprintf(fp,"\n");
}
fprintf(fp,"\n\n");
flushall();
for (i=0;i<n-1;i++)
{
  max = abs(a[i][i]);
  max_row = i;
  for (j=i;j<n;j++)
  {
    if(abs(a[j][i]) > max)
    {
      max = a[j][i];
      max_row = j;
      for (k=0;k<=n;k++)
      {
            temp = a[i][k];
            a[i][k] = a[max_row][k];
            a[max_row][k] = temp;
```

```
      }
    }
  }
    pivot = a[i][i];
    if (pivot == 0.0)
    {
err : fprintf(fp,"********* Solution doesn't exist ***********\n");
      flushall();
      fcloseall();
      exit(0);
    }
    for (l=i+1;l<n;l++)
    {
      mul = a[l][i];
      for (j=i;j<=n;j++)
              a[l][j] = a[l][j] - ((a[i][j] * mul)/pivot);
    }
}
  if (a[n-1][n-1] == 0.0)
     goto err;
  ans[n-1] = a[n-1][n]/a[n-1][n-1];
  for (i=n-2;i>=0;i--)
  {
    temp = 0;
    for (j=n-1;j>=i+1;j--)
    temp=temp+(a[i][j]*ans[j])/a[i][i];
    ans[i]=(a[i][n]/a[i][i])-temp;
  }
  fprintf(fp,"\nThe source strengths are found to be :\n\n");
  flushall();
  for (i=0;i<n;i++)
    fprintf(fp,"%s\t:  %f\n",name1[i],ans[i]);
  fclose(fp);
  clrscr();
}
```

```c
                    /********ELM4.C************/

              /**********************************/
             /********MATRIX-ELIMINATION*******/
            /*********FILE - SCRN*************/
           /*********************************/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

void main()
{
int n,i,j,l,k,max_row,k1,k2;
float a[30][30],ans[30],max,temp,pivot,mul;
char  name[30],*name1[30],*name2[30];
FILE  *fp;

clrscr();
gotoxy(10,5);
highvideo();
cprintf("\tM A T R I X   E L I M I N A T I O N\n\n");
gotoxy(10,20);
printf("DON'T GIVE ANY EXTENSION (.ELM ASSUMED)...");
normvideo();
gotoxy(5,10);
printf("Kindly give the file-name : ");
aq : gets(name);
if (strchr(name,'.') != NULL)
  {
  sound(700); delay(100);
  sound(900); delay(100);
  nosound();
  gotoxy(5,15);
  printf("NO EXTENSION PLEASE...!!!!!!");
  delay(1200);
  gotoxy(5,15);
  clreol();
  gotoxy(33,10);
  clreol();
  flushall();
  goto aq;
 }
 strcat(name,".ELM");
 fp= fopen(name,"r");
 if (!fp)
 {
   sound(1000); delay(100);
   sound(1100); delay(100);
   nosound();
   gotoxy(5,15);
   printf("!!!...NO SUCH FILE EXISTS...!!!");
   delay(1200);
   gotoxy(5,15);
   clreol();
```

```
  gotoxy(33,10);
  clreol();
  goto aq;
}
fscanf(fp,"%d",&n);
flushall();
gotoxy(5,10);
for (i=0;i<n;i++)
{
  name1[i]=malloc(30);
  *name1[i]=NULL;
  fscanf(fp,"%s",name);
  strcpy(name1[i],name);
}
flushall();
for (i=0;i<=n;i++)
{
  flushall();
  name2[i]=malloc(30);
  *name2[i]=NULL;
  fscanf(fp,"%s",name);
  strcpy(name2[i],name);
}
flushall();
for (i=0;i<n;i++)
{
   for (j=0;j<=n;j++)
   {
   fscanf(fp,"%f",&a[i][j]);
   }
}
fclose(fp);
flushall();
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
gotoxy(10,10);
printf("The given matrix is :\n");
for (i=0;i<n;i++)
{
  for (j=0;j<=n;j++)
    printf("%f  ",a[i][j]);
  printf("\n");
}
gotoxy(5,23);
highvideo();
cprintf("PRESS ANY KEY...!!!");
normvideo();
getch();
flushall();
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
for (i=0;i<n-1;i++)
{
  max = abs(a[i][i]);
  max_row = i;
  for (j=i;j<n;j++)
  {
```

```
    if(abs(a[j][i]) > max)
    {
      max = a[j][i];
      max_row = j;
      for (k=0;k<=n;k++)
      {
             temp = a[i][k];
             a[i][k] = a[max_row][k];
             a[max_row][k] = temp;
      }
    }
   }
    pivot = a[i][i];
    if (pivot == 0.0)
    {
      gotoxy(10,10);
err : printf("********* Solution doesn't exist ***********\n")
      gotoxy(5,23);
      highvideo();
      cprintf("PRESS ANY KEY...!!!");
      normvideo();
      getch();
      flushall();
      for (i=6;i<25;i++)
      {
             gotoxy(1,i);
             clreol();
      }
      exit(0) ;
    }
    for (l=i+1;l<n;l++)
    {
      mul = a[l][i];
      for (j=i;j<=n;j++)
             a[l][j] = a[l][j] - ((a[i][j] * mul)/pivot);
    }
  }
   if (a[n-1][n-1] == 0.0)
      goto err;
   ans[n-1] = a[n-1][n]/a[n-1][n-1];
   for (i=n-2;i>=0;i--)
   {
     temp = 0;
     for (j=n-1;j>=i+1;j--)
     temp=temp+(a[i][j]*ans[j])/a[i][i];
     ans[i]=(a[i][n]/a[i][i])-temp;
   }
   gotoxy(10,10);
   printf("The source strengths are found to be :\n\n");
   fflush(stdin);
   for (i=0;i<n;i++)
     printf("%s\t:  %f\n",namel[i],ans[i]);
   fcloseall();
   gotoxy(5,23);
   highvideo();
   cprintf("PRESS ANY KEY...!!!");
   normvideo();
   getch();
   flushall();
   for (i=6;i<25;i++)
   {
     gotoxy(1,i);
     clreol();
```

```
 {
 clrscr();
}
```

```
/********ELM5.C************/

          /*********************************/
         /*********MATRIX-ELIMINATION*******/
        /**********SCRN - SCRN**************/
       /*********************************/



#include <stdio.h>
#include <stdlib.h>
#include <conio.h>


void main()
{
int n,i,j,l,k,max_row,k1,k2;
float a[30][30],ans[30],max,temp,pivot,mul;
char  name[30],*name1[30],*name2[30];
FILE  *fp;



clrscr();
gotoxy(10,5);
highvideo();
cprintf("\tM A T R I X    E L I M I N A T I O N\n\n");
gotoxy(10,20);
cprintf("DON'T GIVE ANY EXTENSION (.ELM ASSUMED)...");
normvideo();
gotoxy(5,10);
printf("Kindly give the file-name : ");
aq : gets(name);
if (strchr(name,'.') != NULL)
 {
   sound(700); delay(100);
   sound(900); delay(100);
   nosound();
   gotoxy(5,15);
   printf("NO EXTENSION PLEASE...!!!!!!");
   delay(1200);
   gotoxy(5,15);
   clreol();
   gotoxy(33,10);
   clreol();
   flushall();
   goto aq;
 }
 strcat(name,".ELM");
 fp= fopen(name,"r");
 if (!fp)
 {
   sound(1000); delay(100);
   sound(1100); delay(100);
   nosound();
   gotoxy(5,15);
   printf("!!!...NO SUCH FILE EXISTS...!!!");
   delay(1200);
   gotoxy(5,15);
```

```
  clreol();
  gotoxy(33,10);
  clreol();
  goto aq;
}
fscanf(fp,"%d",&n);
flushall();
gotoxy(5,10);
for (i=0;i<n;i++)
{
  gotoxy(31,10);
  clreol();
  flushall();
  name1[i]=malloc(30);
  *name1[i]=NULL;
  fscanf(fp,"%s",name);
  strcpy(name1[i],name);
}
flushall();
gotoxy(5,10);
for (i=0;i<=n;i++)
{
  flushall();
  gotoxy(32,10);
  clreol();
  name2[i]=malloc(30);
  *name2[i]=NULL;
  fscanf(fp,"%s",name);
  strcpy(name2[i],name);
}
flushall();
gotoxy(10,12);
highvideo();
normvideo();
for (i=0;i<n;i++)
{
   for (j=0;j<=n;j++)
   {
   gotoxy(42,12);
   clreol();
   fscanf(fp,"%f",&a[i][j]);
   }
}
fclose(fp);
fflush(stdin);
fprintf(stdprn,"\nThe given matrix is :\n\n");
fprintf(stdprn,"Species : \t");
for (i=0;i<n;i++)
  fprintf(stdprn,"%s\t\t",name2[i]);
fprintf(stdprn,"\n\n\n");
for (i=0;i<n;i++)
{
  fprintf(stdprn,"%s\t",name1[i]);
  for (j=0;j<=n;j++)
    fprintf(stdprn,"%f\t",a[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");
flushall();
for (i=0;i<n-1;i++)
{
  max = abs(a[i][i]);
  max_row = i;
```

```
 for (j=i;j<n;j++)
 {
   if(abs(a[j][i]) > max)
   {
     max = a[j][i];
     max_row = j;
     for (k=0;k<=n;k++)
     {
             temp = a[i][k];
             a[i][k] = a[max_row][k];
             a[max_row][k] = temp;
     }
   }
 }
   pivot = a[i][i];
   if (pivot == 0.0)
   {
err : fprintf(stdprn,"********* Solution doesn't exist ***********\n");
     flushall();
     exit(0) ;
   }
   for (l=i+1;l<n;l++)
   {
     mul = a[l][i];
     for (j=i;j<=n;j++)
             a[l][j] = a[l][j] - ((a[i][j] * mul)/pivot);
   }
}
  if (a[n-1][n-1] == 0.0)
     goto err;
  ans[n-1] = a[n-1][n]/a[n-1][n-1];
  for (i=n-2;i>=0;i--)
  {
    temp = 0;
    for (j=n-1;j>=i+1;j--)
    temp=temp+(a[i][j]*ans[j])/a[i][i];
    ans[i]=(a[i][n]/a[i][i])-temp;
  }
  highvideo();
  fprintf(stdprn,"\nThe source strengths are found to be :\n\n");
  normvideo();
  flushall();
  for (i=0;i<n;i++)
    fprintf(stdprn,"%s\t:  %f\n",name1[i],ans[i]);
  clrscr();
 }
```

```
/********ELM6.C************/

    /**********************************/
   /********MATRIX-ELIMINATION*******/
  /*********SCRN - SCRN*************/
 /*********************************/


#include <stdio.h>
#include <stdlib.h>
#include <conio.h>


void main()
{
int n,i,j,l,k,max_row,k1,k2;
float a[30][30],ans[30],max,temp,pivot,mul;
char  c,name[30],*name1[30],*name2[30];
FILE  *fp,*fp1;


 clrscr();
 gotoxy(10,5);
 highvideo();
 cprintf("\tM A T R I X   E L I M I N A T I O N\n\n");
 gotoxy(10,20);
 cprintf("DON'T GIVE ANY EXTENSION (.ELM ASSUMED)...");
 normvideo();
 gotoxy(5,10);
 printf("Kindly give the file-name : ");
 aq : gets(name);
 if (strchr(name,'.') != NULL)
   {
   sound(700); delay(100);
   sound(900); delay(100);
   nosound();
   gotoxy(5,15);
   printf("NO EXTENSION PLEASE...!!!!!!");
   delay(1200);
   gotoxy(5,15);
   clreol();
   gotoxy(33,10);
   clreol();
   flushall();
   goto aq;
  }
  strcat(name,".ELM");
  fp= fopen(name,"r");
  if (!fp)
  {
    sound(1000); delay(100);
    sound(1100); delay(100);
    nosound();
    gotoxy(5,15);
    printf("!!!...NO SUCH FILE EXISTS...!!!");
    delay(1200);
    gotoxy(5,15);
```

```
  clreol();
  gotoxy(33,10);
  clreol();
  goto aq;
}
a  :  gotoxy(5,7);
printf("Kindly give the name of the output file :  ");
gets(name);
fp1 = fopen(name,"r");
if (fp1)
{
  gotoxy(5,15);
  highvideo();
  cprintf("FILE ALREADY EXISTS ...!!!...");
  gotoxy(5,17);
  cprintf("\tOVERWRITE (Y/N) ....????");
b : c=getch();
  switch(c)
  {
    case 'y' :
    case 'Y' : break;
    case 'n' :
    case 'N' : {
                                        gotoxy(5,17);
                                        clreol();
                                        gotoxy(5,15);
                                        clreol();
                                        fcloseall();
                                        flushall();
                                        goto a;
                                   }
    default        : {
                                        sound(100); delay(100);
                                        sound(200); delay(100);
                                        nosound();
                                        goto b;
                                   }
   }
 }
 fclose(fp1);
 fp1 = fopen(name,"w");
 if (!fp1)
 {
   gotoxy(5,15);
   highvideo();
   cprintf("Can't open the file ...!!!...");
   delay(1500);
   gotoxy(5,15);
   clreol();
   normvideo();
   goto a;
 }
 fscanf(fp,"%d",&n);
 flushall();
 gotoxy(5,10);
 for (i=0;i<n;i++)
 {
   gotoxy(31,10);
   clreol();
   flushall();
   name1[i]=malloc(30);
   *name1[i]=NULL;
   fscanf(fp,"%s",name);
```

```
  strcpy(name1[i],name);
}
flushall();
gotoxy(5,10);
for (i=0;i<=n;i++)
{
  flushall();
  gotoxy(32,10);
  clreol();
  name2[i]=malloc(30);
  *name2[i]=NULL;
  fscanf(fp,"%s",name);
  strcpy(name2[i],name);
}
flushall();
gotoxy(10,12);
highvideo();
normvideo();
for (i=0;i<n;i++)
{
   for (j=0;j<=n;j++)
   {
   gotoxy(42,12);
   clreol();
   fscanf(fp,"%f",&a[i][j]);
   }
}
fclose(fp);
fflush(stdin);
fprintf(fp1,"\nThe given matrix is :\n");
fprintf(fp1,"Species : \t");
for (i=0;i<n;i++)
  fprintf(fp1,"%s\t\t",name2[i]);
fprintf(fp1,"\n\n");
for (i=0;i<n;i++)
{
  fprintf(fp1,"%s\t",name1[i]);
  for (j=0;j<=n;j++)
    fprintf(fp1,"%f\t",a[i][j]);
  fprintf(fp1,"\n");
}
fprintf(fp1,"\n\n");
flushall();
for (i=0;i<n-1;i++)
{
  max = abs(a[i][i]);
  max_row = i;
  for (j=i;j<n;j++)
  {
    if(abs(a[j][i]) > max)
    {
      max = a[j][i];
      max_row = j;
      for (k=0;k<=n;k++)
      {
             temp = a[i][k];
             a[i][k] = a[max_row][k];
             a[max_row][k] = temp;
      }
    }
  }
    pivot = a[i][i];
    if (pivot == 0.0)
```

```
    {
err : fprintf(fp1,"********* Solution doesn't exist ***********\n");
      flushall();
      fcloseall();
      exit(0);
    }
    for (l=i+1;l<n;l++)
    {
      mul = a[l][i];
      for (j=i;j<=n;j++)
            a[l][j] = a[l][j] - ((a[i][j] * mul)/pivot);
    }
 }
  if (a[n-1][n-1] == 0.0)
     goto err;
  ans[n-1] = a[n-1][n]/a[n-1][n-1];
  for (i=n-2;i>=0;i--)
  {
    temp = 0;
    for (j=n-1;j>=i+1;j--)
    temp=temp+(a[i][j]*ans[j])/a[i][i];
    ans[i]=(a[i][n]/a[i][i])-temp;
  }
  highvideo();
  fprintf(fp1,"\nThe source strengths are found to be :\n\n");
  normvideo();
  flushall();
  for (i=0;i<n;i++)
    fprintf(fp1,"%s\t:  %f\n",name1[i],ans[i]);
  fclose(fp1);
  clrscr();
 }
```

```
/* FACTOR */

#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "transpose.c"
#include "mat_mul.c"
#define  MAX  50

int factor(float r[MAX][MAX],int nsp,float d[MAX][MAX],float fact[MAX][MAX],float var[MAX])
{
  int   n1,n2,i,j,i1,j1,iteration=1,count=0;

  float b[MAX][MAX],commun[MAX],b12[MAX][MAX],tempo[MAX][MAX],
           b1[MAX][MAX],b2[MAX][MAX],r1[MAX][MAX],
           epslon=.00001,eps1=0.0001,sum12,sum22,sum13,theta;




  for (i=0;i<nsp;i++)
    for (j=0;j<nsp;j++)
      d[i][j] = r[i][j];


    sum12 = 0.0;
    for (i1=0;i1<nsp;i1++)
      sum12 = sum12+d[i1][i1];


   for (i=0;i<nsp;i++)
     for (j=0;j<nsp;j++)
       if (i == j)
              b[i][j] = 1.0;
       else b[i][j] = 0.0;


   do
   {
   gotoxy(1,wherey());
     for (i=0;i<nsp-1;i++)
     {
       for (j=i+1;j<nsp;j++)
       {
              if (d[i][i]-d[j][j] == 0.0)
              {
                printf("\nDivision by Zero...!");
                goto out;
              }
              if (fabs(d[i][j]) <= epslon)  goto bye;
              theta = atan(-2.0*d[i][j]/(d[i][i]-d[j][j]))/2.0;
              for (i1=0;i1<nsp;i1++)
                for (j1=0;j1<nsp;j1++)
                {
                  if (i1==i && j1==i)
                    b1[i1][j1] = cos(theta);
                  else
                    if (i1==i && j1==j)
                           b1[i1][j1] = -sin(theta);
                    else
                           if (i1==j && j1==i)
                             b1[i1][j1] = sin(theta);
```

```
                    else
                      if (i1==j && j1==j)
                        b1[i1][j1] = cos(theta);
                      else
                        if (i1==j1)
                          b1[i1][j1] = 1.0;
                        else
                          b1[i1][j1] = 0.0;
             }
           if (mat_mul(b1,nsp,nsp,b,nsp,nsp,b2)==0) goto out;
           for (i1=0;i1<nsp;i1++)
             for (j1=0;j1<nsp;j1++)
                 b[i1][j1]=b2[i1][j1];
           transpose(b1,nsp,nsp,b12);
           if (mat_mul(b1,nsp,nsp,d,nsp,nsp,tempo)==0) goto out;
           if (mat_mul(tempo,nsp,nsp,b12,nsp,nsp,d)==0) goto out;
bye: ;
      }
    }
    sum22 = 0.0;
    for (i1=0;i1<nsp;i1++)
      sum22 = sum22+d[i1][i1];
    if ((fabs(sum22 - sum12)) > eps1)
    {
      printf("\n|Trace(D) - Trace(R)| > %f \n",eps1);
      goto a1;
    }
    count = 0;
      for (i1=0;i1<nsp-1;i1++)
              for (j1=i1+1;j1<nsp;j1++)
                if (d[i1][j1] <= epslon) count++;
    printf("iteration = %d, count = %d",iteration,2*count);
    delay(200);
    if (count == nsp*(nsp-1)/2)
    {
       printf("\nNo. of off-diagonal elements approaching zero %d\n",2*count);
       goto a1;
    }
    iteration = iteration+1;
  }
  while(iteration <= 5000);



 a1:  n1 = 0;
   for (i=0;i<nsp;i++)
   {
     if (d[i][i] >= .3)
     {

       for (j=0;j<nsp;j++)
               fact[j][n1] = b[i][j] * (sqrt(d[i][i]));
       n1 = n1 + 1;
     }
   }

   for (j=0;j<n1;j++)
   {
     sum13 = 0.0;
     for (i=0;i<nsp;i++)
       sum13 = sum13 + (fact[i][j]*fact[i][j]);
     var[j] = sum13;
   }
```

```
.
  return(nl);
out: return(-1);
|
```

```
            /********** LSQ1.C **********/



#include <stdio.h>
#include "transpos.c"
#include "mat_inv.c"
#include "mat_mul.c"
#define MAX  50

                /********** LEAST SQUARES METHOD *************/

void main()
{


float c[MAX][MAX],a[MAX][MAX],sigma[MAX],temp1[MAX][MAX],temp2[MAX][MAX],temp3[MAX][MAX];
float a_trans[MAX][MAX],temp2_inv[MAX][MAX],w[MAX][MAX],s[MAX][MAX];
int  p,n,i,il,j;
char  *spc_name[30],*src_name[30],name[30];

  clrscr();
  gotoxy(5,5);
  highvideo();
  cprintf("L E A S T - S Q U A R E S   M E T H O D");
  normvideo();
  gotoxy(5,15);
  printf("Kindly give the no. of sources : ");
  scanf("%d",&p);
  gotoxy(5,20);
  printf("Kindly give the no. of species :");
  scanf("%d",&n);
  flushall();
  gotoxy(5,20);
  clreol();
  gotoxy(5,15);
  clreol();
  printf("Name of the next source ? ");
  for (i=0;i<p;i++)
  {
    src_name[i] = malloc(30);
    *src_name[i] = NULL;
    gets(name);
    strcpy(src_name[i],name);
    flushall();
    gotoxy(31,15);
    clreol();
  }
  flushall();
  gotoxy(5,15);
  clreol();
  printf("Name of the next species ? ");
  for (i=0;i<n;i++)
  {
    spc_name[i] = malloc(30);
    *spc_name[i] = NULL;
    gets(name);
    strcpy(spc_name[i],name);
    flushall();
    gotoxy(32,15);
    clreol();
  }
  flushall();
```

```
gotoxy(5,15);
clreol();
gotoxy(5,17);
printf("Please input the concentrations observed for ");
for (i=0;i<n;i++)
{
  gotoxy(50,17);
  printf("%s : ",spc_name[i]);
  scanf("%f",&c[i][0]);
  flushall();
  gotoxy(50,17);
  clreol();
}
flushall();
gotoxy(5,17);
clreol();
gotoxy(5,20);
printf("Please input the source compositions : ");
for (i=0;i<n;i++)
   for (j=0;j<p;j++)
   {
     scanf("%f",&a[i][j]);
     flushall();
     gotoxy(44,20);
     clreol();
   }
flushall();
gotoxy(5,20);
clreol();
gotoxy(5,17);
printf("Kindly give the uncertainties in the measurement of ");
for (i=0;i<n;i++)
{
  printf("%s : ",spc_name[i]);
  scanf("%f",&sigma[i]);
  flushall();
  gotoxy(57,17);
  clreol();
}
flushall();
gotoxy(5,17);
clreol();
for (i=0;i<n;i++)
   for (j=0;j<n;j++)
            if (i==j)
               w[i][j] = 1.0/(sigma[i]*sigma[i]);
            else
               w[i][j] = 0.0 ;
transpose(a,n,p,a_trans);
if (mat_mul(a_trans,p,n,w,n,n,temp1)==0)
  goto out;
if (mat_mul(temp1,p,n,a,n,p,temp2)==0)
  goto out;
if (mat_inv(temp2,p,temp2_inv)==0)
  goto out;
if (mat_mul(temp1,p,n,c,n,1,temp3)==0)
  goto out;
if (mat_mul(temp2_inv,p,p,temp3,p,1,s)==0)
  goto out;
gotoxy(5,10);
printf("The Source Strengths are :");
printf("\n\n");
for (i=0;i<p;i++)
```

```
    printf("%s\t: %f\n",src_name[i],s[i][0]);
  printf("\n\n");
  highvideo();
  cprintf("PRESS ANY KEY...!!!");
  normvideo();
  getch();
  flushall();
out : ;
}
```

```
/********** LSQ2.C **********/

#include <stdio.h>
#include "transpos.c"
#include "mat_inv.c"
#include "mat_mul.c"
#define MAX  50

          /********** LEAST SQUARES METHOD *************/

void main()
{

float c[MAX][MAX],a[MAX][MAX],sigma[MAX],temp1[MAX][MAX],temp2[MAX][MAX],temp3[MAX][MAX];
float a_trans[MAX][MAX],temp2_inv[MAX][MAX],w[MAX][MAX],s[MAX][MAX];
int   p,n,i,i1,j;
char  *spc_name[30],*src_name[30];

  clrscr();
  gotoxy(5,5);
  highvideo();
  cprintf("L E A S T - S Q U A R E S   M E T H O D");
  normvideo();
  gotoxy(5,15);
  printf("Kindly give the no. of sources : ");
  scanf("%d",&p);
  gotoxy(5,20);
  printf("Kindly give the no. of species : ");
  scanf("%d",&n);
  flushall();
  gotoxy(5,20);
  clreol();
  gotoxy(5,15);
  clreol();
  printf("Name of the next source ? ");
  for (i=0;i<p;i++)
  {
    src_name[i] = malloc(30);
    *src_name[i] = NULL;
    gets(src_name[i]);
    flushall();
    gotoxy(31,15);
    clreol();
  }
  flushall();
  gotoxy(5,15);
  clreol();
  printf("Name of the next species ? ");
  for (i=0;i<n;i++)
  {
    spc_name[i] = malloc(30);
    *spc_name[i] = NULL;
    gets(spc_name[i]);
    flushall();
    gotoxy(32,15);
    clreol();
  }
  flushall();
  gotoxy(5,15);
  clreol();
```

```
gotoxy(5,17);
printf("Please input the concentrations observed for ");
for (i=0;i<n;i++)
{
  gotoxy(50,17);
  printf("%s : ",spc_name[i]);
  scanf("%f",&c[i][0]);
  flushall();
  gotoxy(50,17);
  clreol();
}
flushall();
gotoxy(5,17);
clreol();
gotoxy(5,20);
printf("Please input the source compositions : ");
for (i=0;i<n;i++)
   for (j=0;j<p;j++)
   {
     scanf("%f",&a[i][j]);
     flushall();
     gotoxy(44,20);
     clreol();
   }
flushall();
gotoxy(5,20);
clreol();
gotoxy(5,17);
printf("Kindly give the uncertainties in the measurement of ");
for (i=0;i<n;i++)
{
  printf("%s : ",spc_name[i]);
  scanf("%f",&sigma[i]);
  flushall();
  gotoxy(57,17);
  clreol();
}
flushall();
gotoxy(5,17);
clreol();
for (i=0;i<n;i++)
   for (j=0;j<n;j++)
           if (i==j)
              w[i][j] = 1.0/(sigma[i]*sigma[i]);
           else
              w[i][j] = 0.0 ;
transpose(a,n,p,a_trans);
if (mat_mul(a_trans,p,n,w,n,n,temp1)==0)
  goto out;
if (mat_mul(temp1,p,n,a,n,p,temp2)==0)
  goto out;
if (mat_inv(temp2,p,temp2_inv)==0)
  goto out;
if (mat_mul(temp1,p,n,c,n,1,temp3)==0)
  goto out;
if (mat_mul(temp2_inv,p,p,temp3,p,1,s)==0)
  goto out;
flushall();
fprintf(stdprn,"\tL E A S T -- S Q U A R E  M E T H O D\n\n\n");
fprintf(stdprn,"\n\nThe sources are :\n\n");
for (i=0;i<p;i++)
  fprintf(stdprn,"%s  ",src_name[i]);
fprintf(stdprn,"\n\nThe species are :\n\n");
```

```
  for (i=0;i<n;i++)
    fprintf(stdprn,"%s  ",spc_name[i]);
  fprintf(stdprn,"\n\n");
  for (i=0;i<n;i++)
    fprintf(stdprn,"\nThe concentration observed for %s  : %f\n",spc_name[i],c[i][0]);
  fprintf(stdprn,"\n\n");
  for (i=0;i<n;i++)
    fprintf(stdprn,"\nThe uncertainty in the measurement of %s  : %f\n",spc_name[i],sigma[i])
  fprintf(stdprn,"\n\n\nThe given source-composition matrix is :\n\n");
  for (i=0;i<n;i++)
  {
     for (j=0;j<p;j++)
       fprintf(stdprn,"%f  ",a[i][j]);
     fprintf(stdprn,"\n");
  }
  fprintf(stdprn,"\n\nThe Source Strengths are :\n\n");
  for (i=0;i<p;i++)
     fprintf(stdprn,"%s\t: %f\n",src_name[i],s[i][0]);
out : ;
}
```

```
/********** LSQ3.C **********/



#include <stdio.h>
#include "transpos.c"
#include "mat_inv.c"
#include "mat_mul.c"
#define MAX  50

             /********** LEAST SQUARES METHOD *************/

void main()
{


float c[MAX][MAX],a[MAX][MAX],sigma[MAX],temp1[MAX][MAX],temp2[MAX][MAX],temp3[MAX][MAX];
float a_trans[MAX][MAX],temp2_inv[MAX][MAX],w[MAX][MAX],s[MAX][MAX];
int   p,n,i,i1,j;
char  c1,*spc_name[30],*src_name[30],name[30];
FILE  *fp;


  clrscr();
  gotoxy(5,5);
  highvideo();
  cprintf("L E A S T - S Q U A R E S   M E T H O D");
  normvideo();
a:  gotoxy(5,7);
  printf("Kindly give the name of the output file :  ");
  gets(name);
  fp = fopen(name,"r");
  if (fp)
  {
    gotoxy(5,15);
    highvideo();
    cprintf("FILE ALREADY EXISTS ...!!!...");
    gotoxy(5,17);
    cprintf("\tOVERWRITE (Y/N) ....????");
    normvideo();
 b: c1=getch();
    switch(c1)
    {
      case 'y' :
      case 'Y' : break;
      case 'n' :
      case 'N' : {
                                  gotoxy(5,17);
                                  clreol();
                                  gotoxy(5,15);
                                  clreol();
                                  fcloseall();
                                  flushall();
                                  gotoxy(48,7);
                                  clreol();
                                  goto a;
                        }
      default  : {
                                  sound(100); delay(100);
                                  sound(200); delay(100);
                                  nosound();
                                  goto b;
                        }
```

```
  }
}
fclose(fp);
gotoxy(5,17);
clreol();
gotoxy(5,15);
clreol();
fp = fopen(name,"w");
if (!fp)
{
  gotoxy(5,15);
  highvideo();
  cprintf("Can't open the file ...!!!...");
  delay(1500);
  gotoxy(5,15);
  clreol();
  gotoxy(48,7);
  clreol();
  normvideo();
  goto a;
}
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
gotoxy(5,15);
printf("Kindly give the no. of sources : ");
scanf("%d",&p);
gotoxy(5,20);
printf("Kindly give the no. of species : ");
scanf("%d",&n);
flushall();
gotoxy(5,20);
clreol();
gotoxy(5,15);
clreol();
printf("Name of the next source ? ");
for (i=0;i<p;i++)
{
  src_name[i] = malloc(30);
  *src_name[i] = NULL;
  gets(src_name[i]);
  flushall();
  gotoxy(31,15);
  clreol();
}
flushall();
gotoxy(5,15);
clreol();
printf("Name of the next species ? ");
for (i=0;i<n;i++)
{
  spc_name[i] = malloc(30);
  *spc_name[i] = NULL;
  gets(spc_name[i]);
  flushall();
  gotoxy(32,15);
  clreol();
}
flushall();
gotoxy(5,15);
clreol();
```

```
gotoxy(5,17);
printf("Please input the concentrations observed for ");
for (i=0;i<n;i++)
{
  gotoxy(50,17);
  printf("%s : ",spc_name[i]);
  scanf("%f",&c[i][0]);
  flushall();
  gotoxy(50,17);
  clreol();
}
flushall();
gotoxy(5,17);
clreol();
gotoxy(5,20);
printf("Please input the source compositions : ");
for (i=0;i<n;i++)
   for (j=0;j<p;j++)
   {
     scanf("%f",&a[i][j]);
     flushall();
     gotoxy(44,20);
     clreol();
   }
flushall();
gotoxy(5,20);
clreol();
gotoxy(5,17);
printf("Kindly give the uncertainties in the measurement of ");
for (i=0;i<n;i++)
{
  printf("%s : ",spc_name[i]);
  scanf("%f",&sigma[i]);
  flushall();
  gotoxy(57,17);
  clreol();
}
flushall();
gotoxy(5,17);
clreol();
for (i=0;i<n;i++)
   for (j=0;j<n;j++)
          if (i==j)
             w[i][j] = 1.0/(sigma[i]*sigma[i]);
          else
             w[i][j] = 0.0 ;
transpose(a,n,p,a_trans);
if (mat_mul(a_trans,p,n,w,n,n,templ)==0)
  goto out;
if (mat_mul(templ,p,n,a,n,p,temp2)==0)
  goto out;
if (mat_inv(temp2,p,temp2_inv)==0)
  goto out;
if (mat_mul(templ,p,n,c,n,1,temp3)==0)
  goto out;
if (mat_mul(temp2_inv,p,p,temp3,p,1,s)==0)
  goto out;
flushall();
fprintf(fp,"\tL E A S T -- S Q U A R E  M E T H O D\n\n\n");
fprintf(fp,"\n\nThe sources are :\n\n");
for (i=0;i<p;i++)
  fprintf(fp,"%s  ",src_name[i]);
fprintf(fp,"\n\nThe species are :\n\n");
```

```
  for (i=0;i<n;i++)
    fprintf(fp,"%s  ",spc_name[i]);
  fprintf(fp,"\n\n");
  for (i=0;i<n;i++)
    fprintf(fp,"\nThe concentration observed for %s  : %f\n",spc_name[i],c[i][0]);
  fprintf(fp,"\n\n");
  for (i=0;i<n;i++)
    fprintf(fp,"\nThe uncertainty in the measurement of %s  : %f\n",spc_name[i],sigma[i]);
  fprintf(fp,"\n\n\nThe given source-composition matrix is :\n\n");
  for (i=0;i<n;i++)
  {
     for (j=0;j<p;j++)
       fprintf(fp,"%f  ",a[i][j]);
     fprintf(fp,"\n");
  }
  fprintf(fp,"\n\nThe Source Strengths are :\n\n");
  for (i=0;i<p;i++)
     fprintf(fp,"%s\t: %f\n",src_name[i],s[i][0]);
out : ;
}
```

```
/********** LSQ4.C **********/



#include <stdio.h>
#include "transpos.c"
#include "mat_inv.c"
#include "mat_mul.c"
#define MAX  50

          /********* LEAST SQUARES METHOD *************/

void main()
{


float c[MAX][MAX],a[MAX][MAX],sigma[MAX],temp1[MAX][MAX],temp2[MAX][MAX],temp3[MAX][MAX];
float a_trans[MAX][MAX],temp2_inv[MAX][MAX],w[MAX][MAX],s[MAX][MAX];
int  p,n,i,il,j;
char  *spc_name[30],*src_name[30],name[30];
FILE  *fp;


  clrscr();
  gotoxy(5,5);
  highvideo();
  cprintf("L E A S T - S Q U A R E S  M E T H O D");
  normvideo();
  gotoxy(10,20);
  cprintf("DON'T GIVE ANY EXTENSION (.LSQ ASSUMED)...");
  normvideo();
  gotoxy(5,10);
  printf("Kindly give the file-name : ");
 aq : gets(name);
  if (strchr(name,'.') != NULL)
  {
    sound(700); delay(100);
    sound(900); delay(100);
    nosound();
    gotoxy(5,15);
    printf("NO EXTENSION PLEASE...!!!!!!");
    delay(1200);
    gotoxy(5,15);
    clreol();
    gotoxy(33,10);
    clreol();
    flushall();
    goto aq;
   }
   strcat(name,".LSQ");
   fp= fopen(name,"r");
   if (!fp)
   {
     sound(1000); delay(100);
     sound(1100); delay(100);
     nosound();
     gotoxy(5,15);
     printf("!!!...NO SUCH FILE EXISTS...!!!");
     delay(1200);
     gotoxy(5,15);
     clreol();
     gotoxy(33,10);
     clreol();
```

```
  goto aq;
}
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
fscanf(fp,"%d",&p);
fscanf(fp,"%d",&n);
flushall();
for (i=0;i<p;i++)
{
  src_name[i] = malloc(30);
  *src_name[i] = NULL;
  fscanf(fp,"%s",name);
  strcpy(src_name[i],name);
  flushall();
}
for (i=0;i<n;i++)
{
  spc_name[i] = malloc(30);
  *spc_name[i] = NULL;
  fscanf(fp,"%s",name);
  strcpy(spc_name[i],name);
  flushall();
}
for (i=0;i<n;i++)
{
  fscanf(fp,"%f",&c[i][0]);
  flushall();
}
for (i=0;i<n;i++)
   for (j=0;j<p;j++)
   {
     fscanf(fp,"%f",&a[i][j]);
     flushall();
   }
for (i=0;i<n;i++)
{
  fscanf(fp,"%f",&sigma[i]);
  flushall();
}
fclose(fp);
for (i=0;i<n;i++)
   for (j=0;j<n;j++)
            if (i==j)
               w[i][j] = 1.0/(sigma[i]*sigma[i]);
            else
               w[i][j] = 0.0 ;
transpose(a,n,p,a_trans);
if (mat_mul(a_trans,p,n,w,n,n,temp1)==0)
  goto out;
if (mat_mul(temp1,p,n,a,n,p,temp2)==0)
  goto out;
if (mat_inv(temp2,p,temp2_inv)==0)
  oto out;
  (mat_mul(temp1,p,n,c,n,1,temp3)==0)
  oto out;
  (mat_mul(temp2_inv,p,p,temp3,p,1,s)==0)
  oto out;
  (i=6;i<25;i++)

  otoxy(1,i);
```

```
   clreol();
  }
  flushall();
  gotoxy(5,10);
  printf("The sources are :\n\n");
  for (i=0;i<p;i++)
    printf("%s  ",src_name[i]);
  gotoxy(5,15);
  printf("The species are :\n\n");
  for (i=0;i<n;i++)
    printf("%s  ",spc_name[i]);
  gotoxy(10,23);
  highvideo();
  cprintf("PRESS ANY KEY ...!!!");
  normvideo();
  getch();
  flushall();
  for (i=6;i<25;i++)
  {
    gotoxy(1,i);
    clreol();
  }
  for (i=0;i<n;i++)
  {
    gotoxy(1,7+i);
    printf("The concentration observed for %s  : %f\n",spc_name[i],c[i][0]);
  }
  gotoxy(10,23);
  highvideo();
  cprintf("PRESS ANY KEY ...!!!");
  normvideo();
  getch();
  for (i=6;i<25;i++)
  {
    gotoxy(1,i);
    clreol();
  }
  for (i=0;i<n;i++)
  {
    gotoxy(1,7+i);
    printf("The uncertainty in the measurement of %s  : %f\n",spc_name[i],sigma[i]);
  }
  gotoxy(10,23);
  highvideo();
  cprintf("PRESS ANY KEY ...!!!");
  normvideo();
  getch();
  for (i=6;i<25;i++)
  {
    gotoxy(1,i);
    clreol();
  }
  gotoxy(10,10);
  printf("The given source-composition matrix is :\n\n");
  for (i=0;i<n;i++)
  {
     for (j=0;j<p;j++)
       printf("%f  ",a[i][j]);
     printf("\n");
  }
  gotoxy(10,23);
  highvideo();
  cprintf("PRESS ANY KEY ...!!!");
```

```
    normvideo();
    getch();
    for (i=6;i<25;i++)
    {
      gotoxy(1,i);
      clreol();
    }
    gotoxy(5,15);
    printf("The Source Strengths are :");
    printf("\n\n");
    for (i=0;i<p;i++)
       printf("%s\t: %f\n",src_name[i],s[i][0]);
    gotoxy(10,23);
    highvideo();
    cprintf("PRESS ANY KEY ...!!!");
    normvideo();
    getch();
    flushall();
  out : ;
  }
```

```
            /********** LSQ5.C **********/



#include <stdio.h>
#include "transpos.c"
#include "mat_inv.c"
#include "mat_mul.c"
#define MAX  50

                /********** LEAST SQUARES METHOD *************/

void main()
{


float c[MAX][MAX],a[MAX][MAX],sigma[MAX],temp1[MAX][MAX],temp2[MAX][MAX],temp3[MAX][MAX];
float a_trans[MAX][MAX],temp2_inv[MAX][MAX],w[MAX][MAX],s[MAX][MAX];
int   p,n,i,i1,j;
char  *spc_name[30],*src_name[30],name[30];
FILE  *fp;


  clrscr();
  gotoxy(5,5);
  highvideo();
  cprintf("L E A S T - S Q U A R E S   M E T H O D");
  normvideo();
  gotoxy(10,20);
  cprintf("DON'T GIVE ANY EXTENSION (.LSQ ASSUMED)...");
  normvideo();
  gotoxy(5,10);
  printf("Kindly give the file-name : ");
aq : gets(name);
  if (strchr(name,'.') != NULL)
  {
    sound(700); delay(100);
    sound(900); delay(100);
    nosound();
    gotoxy(5,15);
    printf("NO EXTENSION PLEASE...!!!!!!");
    delay(1200);
    gotoxy(5,15);
    clreol();
    gotoxy(33,10);
    clreol();
    flushall();
    goto aq;
  }
  strcat(name,".LSQ");
  fp= fopen(name,"r");
  if (!fp)
  {
    sound(1000); delay(100);
    sound(1100); delay(100);
    nosound();
    gotoxy(5,15);
    printf("!!!...NO SUCH FILE EXISTS...!!!");
    delay(1200);
    gotoxy(5,15);
    clreol();
    gotoxy(33,10);
    clreol();
```

```
  goto aq;
}
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
fscanf(fp,"%d",&p);
fscanf(fp,"%d",&n);
flushall();
for (i=0;i<p;i++)
{
  src_name[i] = malloc(30);
  *src_name[i] = NULL;
  fscanf(fp,"%s",name);
  strcpy(src_name[i],name);
  flushall();
}
for (i=0;i<n;i++)
{
  spc_name[i] = malloc(30);
  *spc_name[i] = NULL;
  fscanf(fp,"%s",name);
  strcpy(spc_name[i],name);
  flushall();
}
for (i=0;i<n;i++)
{
  fscanf(fp,"%f",&c[i][0]);
  flushall();
}
for (i=0;i<n;i++)
   for (j=0;j<p;j++)
   {
     fscanf(fp,"%f",&a[i][j]);
     flushall();
   }
for (i=0;i<n;i++)
{
  fscanf(fp,"%f",&sigma[i]);
  flushall();
}
fclose(fp);
for (i=0;i<n;i++)
   for (j=0;j<n;j++)
           if (i==j)
              w[i][j] = 1.0/(sigma[i]*sigma[i]);
           else
              w[i][j] = 0.0 ;
transpose(a,n,p,a_trans);
if (mat_mul(a_trans,p,n,w,n,n,temp1)==0)
  goto out;
if (mat_mul(temp1,p,n,a,n,p,temp2)==0)
  goto out;
if (mat_inv(temp2,p,temp2_inv)==0)
  goto out;
if (mat_mul(temp1,p,n,c,n,1,temp3)==0)
  goto out;
if (mat_mul(temp2_inv,p,p,temp3,p,1,s)==0)
  goto out;
gotoxy(10,23);
highvideo();
cprintf("PRESS ANY KEY ...!!!");
```

```
normvideo();
getch();
flushall();
fprintf(stdprn,"The sources are :\n\n");
for (i=0;i<p;i++)
  fprintf(stdprn,"%s  ",src_name[i]);
fprintf(stdprn,"\n\nThe species are :\n\n");
for (i=0;i<n;i++)
  fprintf(stdprn,"%s  ",spc_name[i]);
gotoxy(10,23);
fprintf(stdprn,"\n\n");
for (i=0;i<n;i++)
  fprintf(stdprn,"\nThe concentration observed for %s  : %f\n",spc_name[i],c[i][0]);
  fprintf(stdprn,"\n\n");
for (i=0;i<n;i++)
  fprintf(stdprn,"\nThe uncertainty in the measurement of %s  : %f\n",spc_name[i],sigma[i]);
fprintf(stdprn,"\n");
fprintf(stdprn,"\n\nThe given source-composition matrix is :\n\n");
for (i=0;i<n;i++)
{
   for (j=0;j<p;j++)
     fprintf(stdprn,"%f  ",a[i][j]);
   fprintf(stdprn,"\n");
}
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
gotoxy(5,15);
fprintf(stdprn,"\n\nThe Source Strengths are :");
fprintf(stdprn,"\n\n");
for (i=0;i<p;i++)
   fprintf(stdprn,"%s\t: %f\n",src_name[i],s[i][0]);
flushall();
out : ;
}
```

```
/********** LSQ6.C **********/


#include <stdio.h>
#include "transpos.c"
#include "mat_inv.c"
#include "mat_mul.c"
#define MAX  50

               /********** LEAST SQUARES METHOD *************/

void main()
{


float c[MAX][MAX],a[MAX][MAX],sigma[MAX],temp1[MAX][MAX],temp2[MAX][MAX],temp3[MAX][MAX];
float a_trans[MAX][MAX],temp2_inv[MAX][MAX],w[MAX][MAX],s[MAX][MAX];
int   p,m,i,i1,j;
char  c1,*spc_name[30],*src_name[30],name[30];
FILE  *fp,*fp1;


  clrscr();
  gotoxy(5,5);
  highvideo();
  cprintf("L E A S T - S Q U A R E S   M E T H O D");
  normvideo();
  gotoxy(10,20);
  cprintf("DON'T GIVE ANY EXTENSION (.LSQ ASSUMED)...");
  gotoxy(5,10);
  printf("Kindly give the input file-name : ");
aq : gets(name);
  if (strchr(name,'.') != NULL)
  {
    sound(700); delay(100);
    sound(900); delay(100);
    nosound();
    gotoxy(5,15);
    printf("NO EXTENSION PLEASE...!!!!!!");
    delay(1200);
    gotoxy(5,15);
    clreol();
    gotoxy(39,10);
    clreol();
    flushall();
    goto aq;
  }
  strcat(name,".LSQ");
  fp= fopen(name,"r");
  if (!fp)
  {
    sound(1000); delay(100);
    sound(1100); delay(100);
    nosound();
    gotoxy(5,15);
    printf("!!!...NO SUCH FILE EXISTS...!!!");
    delay(1200);
    gotoxy(5,15);
    clreol();
    gotoxy(39,10);
    clreol();
    goto aq;
```

```
}
for (i=6;i<25;i++)
{
  gotoxy(1,i);
  clreol();
}
a :  gotoxy(5,7);
 clreol();
 printf("Kindly give the name of the output file :  ");
 gets(name);
 fp1 = fopen(name,"r");
 if (fp1)
 {
   gotoxy(5,15);
   highvideo();
   cprintf("FILE ALREADY EXISTS ...!!!...");
   gotoxy(5,17);
   cprintf("\tOVERWRITE (Y/N) ....????");
   normvideo();
b : c1=getch();
   switch(c1)
   {
     case 'y' :
     case 'Y' : break;
     case 'n' :
     case 'N' : {
                                          gotoxy(5,17);
                                          clreol();
                                          gotoxy(5,15);
                                          clreol();
                                          fcloseall();
                                          flushall();
                                          goto a;
                                    }
    default          : {
                                          sound(100); delay(100);
                                          sound(200); delay(100);
                                          nosound();
                                          goto b;
                                    }
    }
   }
   fclose(fp1);
   fp1 = fopen(name,"w");
   if (!fp1)
   {
     gotoxy(5,15);
     highvideo();
     cprintf("Can't open the file ...!!!...");
     delay(1500);
     gotoxy(5,15);
     clreol();
     normvideo();
     goto a;
   }

   for (i=6;i<25;i++)
   {
     gotoxy(1,i);
     clreol();
   }
   fscanf(fp,"%d",&p);
   fscanf(fp,"%d",&n);
```

```
flushall();
for (i=0;i<p;i++)
{
  src_name[i] = malloc(30);
  *src_name[i] = NULL;
  fscanf(fp,"%s",name);
  strcpy(src_name[i],name);
  flushall();
}
for (i=0;i<n;i++)
{
  spc_name[i] = malloc(30);
  *spc_name[i] = NULL;
  fscanf(fp,"%s",name);
  strcpy(spc_name[i],name);
  flushall();
}
for (i=0;i<n;i++)
{
  fscanf(fp,"%f",&c[i][0]);
  flushall();
}
for (i=0;i<n;i++)
   for (j=0;j<p;j++)
   {
     fscanf(fp,"%f",&a[i][j]);
     flushall();
   }
for (i=0;i<n;i++)
{
  fscanf(fp,"%f",&sigma[i]);
  flushall();
}
fclose(fp);
for (i=0;i<n;i++)
   for (j=0;j<n;j++)
           if (i==j)
              w[i][j] = 1.0/(sigma[i]*sigma[i]);
           else
              w[i][j] = 0.0 ;
transpose(a,n,p,a_trans);
if (mat_mul(a_trans,p,n,w,n,n,temp1)==0)
  goto out;
if (mat_mul(temp1,p,n,a,n,p,temp2)==0)
  goto out;
if (mat_inv(temp2,p,temp2_inv)==0)
  goto out;
if (mat_mul(temp1,p,n,c,n,1,temp3)==0)
  goto out;
if (mat_mul(temp2_inv,p,p,temp3,p,1,s)==0)
  goto out;
flushall();
fprintf(fp1,"\n\nThe sources are :\n\n");
for (i=0;i<p;i++)
  fprintf(fp1,"%s  ",src_name[i]);
fprintf(fp1,"\n\n\nThe species are :\n\n");
for (i=0;i<n;i++)
  fprintf(fp1,"%s  ",spc_name[i]);
fprintf(fp1,"\n\n");
for (i=0;i<n;i++)
  fprintf(fp1,"\nThe concentration observed for %s  : %f\n",spc_name[i],c[i][0]);
fprintf(fp1,"\n\n");
for (i=0;i<n;i++)
```

```
  fprintf(fpl,"\nThe uncertainty in the measurement of %s  : %f\n",spc_name[i],sigma[i]);
 fprintf(fpl,"The given source-composition matrix is :\n\n");
 for (i=0;i<n;i++)
 {
    for (j=0;j<p;j++)
      fprintf(fpl,"%f  ",a[i][j]);
    fprintf(fpl,"\n");
 }
 fprintf(fpl,"\n\nThe Source Strengths are :\n\n");
 fprintf(fpl,"\n\n");
 for (i=0;i<p;i++)
    fprintf(fpl,"%s\t: %f\n",src_name[i],s[i][0]);
 gotoxy(10,23);
 highvideo();
 cprintf("PRESS ANY KEY ...!!!");
 normvideo();
 getch();
 flushall();
out : ;
}
```

```
/********* M A I N . C **************/


/********* M A I N ***********/


#include <stdio.h>
#include <process.h>
#include <alloc.h>

void main()

{
     int decide,temp;


     spawnl(P_WAIT,"menu1.exe",NULL);
     flushall();
a  : decide = 0;
     switch(spawnl(P_WAIT,"menu2.exe",NULL))
     {
            case  '1' : {
                          switch(spawnl(P_WAIT,"menu3.exe",NULL))
                          {
                                 case  '7' : {
                                                decide = 1 ;
                                                break;
                                             }
                          }
                          if (decide == 1)
                                 {
                                  flushall();
                                  fcloseall();
                                  goto a;
                                 }
                          break;
```

```
case '1' : {
                decide = 0;
                temp=spawnl(P_WAIT,"menu5.exe",NULL);
                if(temp=='n'||temp=='N')
                  {
                        flushall();
                        fcloseall();
                        goto  d;
                  }
                switch(spawnl(P_WAIT,"menu6.exe",NULL))
                {
                  case '1' :{
                                switch(temp)
                                {
                                  case 1 :{
                                                spawnl(P_WAIT,"ELM1.exe",NULL);
                                                break;
                                          }
                                  case 2 :{
                                                spawnl(P_WAIT,"ELM2.exe",NULL);
                                                break;
                                          }
                                  case 3 :{
                                                spawnl(P_WAIT,"ELM3.EXE",NULL);
                                                break;
                                          }
                                  case 4 :{
                                                spawnl(P_WAIT,"ELM4.EXE",NULL);
                                                break;
                                          }
                                  case 5 :{
                                                spawnl(P_WAIT,"ELM5.EXE",NULL);
                                                break;
                                          }
                                  case 6 :{
                                                spawnl(P_WAIT,"ELM6.EXE",NULL);
                                                break;
                                          }
```

```
                  flushall();
                  fcloseall();
                  goto b;
                }
case '2' :{
                  switch(temp)
                  {
                    case 1 :{
                                    spawnl(P_WAIT,"LSQ1.EXE",NULL);
                                    break;
                                }
                    case 2 :{
                                    spawnl(P_WAIT,"LSQ2.EXE",NULL);
                                    break;
                                }
                    case 3 :{
                                    spawnl(P_WAIT,"LSQ3.EXE",NULL);
                                    break;
                                }
                    case 4 :{
                                    spawnl(P_WAIT,"LSQ4.EXE",NULL);
                                    break;
                                }
                    case 5 :{
                                    spawnl(P_WAIT,"LSQ5.EXE",NULL);
                                    break;
                                }
                    case 6 :{
                                    spawnl(P_WAIT,"LSQ6.EXE",NULL);
                                    break;
                                }
                  }
                  flushall();
                  fcloseall();
```

```
                              case '3' : {
                                                  decide = 1;
                                                  break;
                                                 }
                             }
                             flushall();
                             fcloseall();
                             if (decide == 1) goto b;
                             break;
                            }
                case '2' : {
c :                          decide = 0;
                             temp=spawnl(P_WAIT,"menu5.exe",NULL);
                             flushall();
                             if(temp=='n'||temp=='N')
                               {
                                      flushall();
                                      fcloseall();
                                      goto  d;
                               }
                             switch(spawnl(P_WAIT,"menu7.exe",NULL))
                             {
                               case '1' :{
                                                 switch(temp)
                                                 {
                                                   case 1 :{
                                                                      spawnl(P_WAIT,"varimax1.exe",NULL);
                                                                      break;
                                                           }
                                                   case 2 :{
                                                                      spawnl(P_WAIT,"varimax2.exe",NULL);
                                                                      break;
                                                           }
                                                   case 3 :{
                                                                      spawnl(P_WAIT,"varimax3.exe",NULL);
                                                                      break;
                                                           }
```

```
                    case 4 :{
                                    spawnl(P_WAIT,"varimax4.exe",NULL);
                                    break;
                                }
                    case 5 :{
                                    spawnl(P_WAIT,"varimax5.exe",NULL);
                                    break;
                                }
                    case 6 :{
                                    spawnl(P_WAIT,"varimax6.exe",NULL);
                                    break;
                                }
                  }
                  flushall();
                  fcloseall();
                  goto c;
                 }
      case '2' :{
                  switch(temp)
                  {
                    case 1 :{
                                    spawnl(P_WAIT,"qmax1.exe",NULL);
                                    break;
                                }
                    case 2 :{
                                    spawnl(P_WAIT,"qmax2.exe",NULL);
                                    break;
                                }
                    case 3 :{
                                    spawnl(P_WAIT,"qmax3.exe",NULL);
                                    break;
                                }
                    case 4 :{
                                    spawnl(P_WAIT,"qmax4.exe",NULL);
                                    break;
                                }
                    case 5 :{
```

```
                                        }
                                    case 6 :{
                                                spawnl(P_WAIT,"qmax6.exe",NULL);
                                                break;
                                        }
                                    }
                                    flushall();
                                    fcloseall();
                                    goto c;
                                   }
                          case '3' : {
                                    decide = 1;
                                    break;
                                   }
                          {
                          fcloseall();
                          flushall();
                          goto c;
                         }
                  case '3' : {
                          decide = 1;
                          break;
                         }
                 }
                 if (decide == 1)
                 {
                       flushall();
                       fcloseall();
                       goto a;
                 }
                 break;
                }
         case  '3'  : break;
      }
   }
```

```
/***********MAT_INV.C************/
/********* M A T R I X   I N V E R S E . C **********/


#include <math.h>
#define MAX 50

int mat_inv(float a1[MAX][MAX],int n,float b[MAX][MAX])
{
int i1,j1,i,j,l,k,max_row,k1,k2;
float pivot,temp,max,mul,a[MAX][MAX];

for (i=0;i<n;i++)
  for (j=0;j<n;j++)
    a[i][j] = a1[i][j];
for (i=0;i<n;i++)
  for (j=n;j<(2*n);j++)
  {
    if ((i+n) == j)
      a[i][j] = 1.0;
    else a[i][j] = 0.0;
  }
 for (i=0;i<n;i++)
 {
  max = abs(a[i][i]);
  max_row = i;
  for (j=i;j<n;j++)
  {
    if(abs(a[j][i]) > max)
    {
      max = a[j][i];
      max_row = j;
      for (k=i;k<(2*n);k++)
      {
              temp = a[i][k];
             a[i][k] = a[max_row][k];
             a[max_row][k] = temp;
      }
    }
  }
    pivot = a[i][i];
    if (pivot == 0)
    {
      printf("********* Inverse doesn't exist ***********");
      return(0) ;
    }
    for (j=0;j<2*n;j++)
      a[i][j] = a[i][j]/pivot ;
    for (l=0;l<n;l++)
    if (l != i)
    {
      mul = a[l][i];
      for (j=i;j<(2*n);j++)
             a[l][j] = a[l][j] - (a[i][j] * mul);
    }
 }
  for (i=0;i<n;i++)
  {
    for (j=n;j<2*n;j++)
    {
    b[i][j-n] = a[i][j];
```

```
    }
  }
return(1);
}
```

```
/********** M E N U 1 . C ************/

/************ M E N U 1 ***************/

#include <graphics.h>
#include <conio.h>
#include <stdlib.h>
void main()
{
   int poly[18]={200,105,190,125,220,110,240,130,230,100,270,70,220,100,200,70,200,105};
   int i,j,gm,gd=DETECT;

   initgraph(&gd,&gm,"");
   cleardevice();
   printf("\a");
   bar3d(50,60,80,250,10,1);   // left vert. line
   bar3d(50,50,600,70,10,1);   // upper horiz. line
   bar3d(50,270,570,250,10,1); // lower horiz. line
   bar3d(570,70,600,270,10,1); // right vert. line
   for (j=0;j<125;j=j+5)
   {
                        circle(80+j,100,8);
                        sound(j); delay(50);
                        floodfill(80+j,100,WHITE);
   }
   drawpoly(9,poly);
   floodfill(201,105,WHITE);
   floodfill(210,105,WHITE);
   for (i=0;i<3;i++)
    {
     sound(1500); delay(500);
     sound(100); delay(500);
    }
    nosound();

  settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
  outtextxy(255,80,"A I R");
  sound(200); delay(50);
  outtextxy(185,130,"Q U A L I T Y");
  sound(300); delay(50);
  outtextxy(155,180,"M O D E L L I N G");
  sound(400); delay(50);
  nosound();
  for (i=0;i<75;i++)
  {
   sound(random(i)*50);
   delay(100);
  }
  nosound();
  closegraph();
```

```
/********* M E N U 2 . C ***********/


/*********** M E N U 2 **************/


#include <graphics.h>
#include <conio.h>
#include <process.h>
#include <alloc.h>

int main()
{
   int c,i,j1,j2,gm,gd=DETECT;
   void *buffer;
   unsigned int size;


b  :  initgraph(&gd,&gm,"");
   cleardevice();
   j1=getbkcolor();
   j2=getcolor();
   rectangle(150,73,500,273);
   sound(50); delay(100);
   pieslice(150,73,270,360,75);
   pieslice(500,73,180,270,75);
   pieslice(500,273,90,180,75);
   pieslice(150,273,0,90,75);
   sound(150); delay(100);
   pieslice(325,73,0,180,100);
   pieslice(325,273,180,360,100);
   pieslice(150,173,270,450,58);
   pieslice(500,173,0,90,58);
   pieslice(500,173,270,360,58);
   sound(75); delay(100); nosound();
   circle(50,150,10);
   circle(50,170,10);
   circle(50,190,10);
   floodfill(50,150,WHITE);
   floodfill(50,170,WHITE);
   floodfill(50,190,WHITE);
   setcolor(j1);
   outtextxy(47,147,"1");
   sound(150); delay(100); nosound();delay(500);
   outtextxy(47,167,"2");
   sound(200); delay(100); nosound(); delay(500);
   outtextxy(47,187,"3");
   sound(250); delay(100); nosound();
   delay(1000);
   size = imagesize(40,140,60,160);
   buffer = (void *)malloc(size);
   getimage(40,140,60,160,(void *)buffer);
   putimage(40,140,(void *)buffer,XOR_PUT);
   putimage(170,140,(void *)buffer,COPY_PUT);
   sound(2000); delay(100); nosound();
   setcolor(j2);
   settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
   outtextxy(250,137,"D A T A"); delay(1000);
   size = imagesize(40,160,60,190);
   buffer = (void *)malloc(size);
   getimage(40,160,60,180,(void *)buffer);
```

```
    putimage(40,160,(void *)buffer,XOR_PUT);
    putimage(170,160,(void *)buffer,COPY_PUT);
    sound(2000); delay(100); nosound();
    setcolor(j2);
    settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
    outtextxy(250,157,"A N A L Y S I S");delay(1000);
    size = imagesize(40,180,60,200);
    buffer = (void *)malloc(size);
    getimage(40,180,60,200,(void *)buffer);
    putimage(40,180,(void *)buffer,XOR_PUT);
    putimage(170,179,(void *)buffer,COPY_PUT);
    sound(2000); delay(100);
    nosound();
    settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
    setcolor(j2);
    outtextxy(250,177,"Q U I T"); delay(1000);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
    setcolor(j1);outtextxy(260,285,"ENTER CHOICE ");
    sound(900);delay(100); nosound(); delay(100);
    sound(900);delay(100); nosound(); delay(100);
    sound(900);delay(100); nosound();
    circle(325,325,10);
    settextstyle(DEFAULT_FONT,HORIZ_DIR,0);
    moveto(323,323);
a :   c = getch();
    switch(c)
    {
      case '1' :
      case '2' :
      case '3' : {
                              sound(300); delay(200);
                              sound(200); delay(200);
                              sound(100); delay(200);
                              nosound();
                              closegraph();
                              exit(c);
                            }
      default  : {
                              sound(100); delay(200);
                              sound(200); delay(200);
                              nosound();
                              goto a ;
                            }
    }
    closegraph();
}
```

/********* M E N U 3 . C **********/

/********* M E N U 3 *********/

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <process.h>


int main()
{
     int gm,gd,j1,j2,i1,i2,c,src1,spc1,i,j,n,n1;
     int poly[8] = {275,55,285,50,285,60,275,55};
     char name[30],name1[30],name2[30],fname1[30],fname2[30],path1[30],path[30],ext[6];
     char *spc_name[30],*src_name[30];
     FILE *fp;
     fpos_t filepos;
     float coeff,conc,uncert,x[50][50];

/********************************************************************/
/*      M E N U                                                    */
/*              S T A R T S                                        */
/*                          H E R E                                */
/********************************************************************/
     j1 = getcolor();
     j2 = getbkcolor();
a :  gd=DETECT;
     initgraph(&gd,&gm,"");
     cleardevice();
     rectangle(150,150,220,220);
     outtextxy(173,165,"D");
     outtextxy(173,175,"A");
     outtextxy(173,185,"T");
     outtextxy(173,195,"A");
     sound(750); delay(200); nosound(); delay(200);
     rectangle(550,20,630,50);
     outtextxy(560,30,"CREATE");
     outtextxy(560,40,"FILE");
     circle(650,35,10);
     outtextxy(647,32,"1");
     line(200,185,550,35);
     line(200,183,550,33);
     sound(1000); delay(200); nosound(); delay(200);
     rectangle(550,70,630,100);
     outtextxy(560,80,"EDIT");
     outtextxy(560,90,"FILE");
     circle(650,85,10);
     outtextxy(647,82,"2");
     line(200,185,550,85);
     line(200,183,550,83);
     sound(1000); delay(200); nosound(); delay(200);
     rectangle(550,120,630,150);
     outtextxy(560,130,"COPY");
     outtextxy(560,140,"FILE");
     circle(650,135,10);
```

```
outtextxy(647,132,"3");
line(200,185,550,135);
line(200,183,550,133);
sound(1000); delay(200); nosound(); delay(200);
rectangle(550,170,630,200);
outtextxy(560,180,"RENAME");
outtextxy(560,190,"FILE");
circle(650,185,10);
outtextxy(647,182,"4");
line(200,185,550,185);
line(200,183,550,183);
sound(1000); delay(200); nosound(); delay(200);
rectangle(550,220,630,250);
outtextxy(560,230,"DELETE");
outtextxy(560,240,"FILE");
circle(650,235,10);
outtextxy(647,232,"5");
line(200,185,550,235);
line(200,183,550,233);
sound(1000); delay(200); nosound(); delay(200);
rectangle(550,270,630,300);
outtextxy(560,280,"LIST OF");
outtextxy(560,290,"FILES");
circle(650,285,10);
outtextxy(647,282,"6");
line(200,185,550,285);
line(200,183,550,283);
sound(1000); delay(200); nosound(); delay(200);
rectangle(550,320,630,345);
outtextxy(560,325,"EXIT");
outtextxy(560,337,"MENU");
circle(650,330,10);
outtextxy(647,327,"7");
line(200,185,550,335);
line(200,183,550,333);
sound(1000); delay(200); nosound(); delay(200);
circle(260,55,10);
drawpoly(4,poly);
rectangle(285,53,295,58);
floodfill(1,1,WHITE);
setcolor(j2);
outtextxy(50,50,"E N T E R   C H O I C E ");
sound(600); delay(200); nosound(); delay(200);
sound(600); delay(200); nosound(); delay(200);
sound(600); delay(200); nosound(); delay(200);
b :  c = getch();
switch(c)
{          ///////////////////////////////
                     /// C R E A T E   F I L E ///
                 ///////////////////////////////

  case  '1': {
                   sound(700); delay(200);
                   sound(800); delay(200);
                   sound(900); delay(200);
                   nosound();

                   restorecrtmode();
                   clrscr();
                   *ext = NULL;
                   gotoxy(10,10);
                   printf("(1) Files used in MATRIX ELIMINATION Method under");
                   gotoxy(10,11);
```

```
                printf("    Chemical Mass Balance shall get an  extension");
                gotoxy(10,12);
                printf("    BLM automatically") ;
                gotoxy(10,14);
                printf("(2) Files used in LEAST SQUARE Method under C.M.B.");
                gotoxy(10,15);
                printf("    shall get an extension LSQ automatically");
                gotoxy(10,17);
                printf("(3) Files used in VARIMAX & QUARTIMAX Rotations ");
                gotoxy(10,18);
                printf("under Factor Analysis shall get an extension FAC");
                gotoxy(10,19);
                printf("automatically");
                gotoxy(10,21);
                printf("(4) To EXIT this menu");
                gotoxy(10,24);
                highvideo();
                cprintf("Kindly enter choice ");
                normvideo();
t  :        c = getch();
                switch(c)
                {
                 case  '1' : {
                                    strcat(ext,".BLM");
                                    break;
                             }
                 case  '2' : {
                                    strcat(ext,".LSQ");
                                    break;
                             }
                 case  '3' : {
                                    strcat(ext,".FAC");
                                    break;
                             }
                 case  '4' :   goto last;
                 default   : {
                                    sound(200); delay(150);
                                    sound(300); delay(150);
                                    nosound();
                                    goto t;
                             }
                }
 cd  :          clrscr();
                gotoxy(10,10);
                *name = NULL;
                printf("Please give the name of the file (don't give extension) : ");
                gets(name);
                if (strstr(name,".") != NULL)
                {
                  gotoxy(10,15);
                  highvideo();
                  printf("No Extension Please...!!!");
                  normvideo();
                  delay(500);
                  gotoxy(10,15);
                  clreol();
                  goto cd;
                }
                if (*name == NULL) goto cd;
                strcat(name,ext);
                fp = fopen(name,"r");
                if (fp != NULL)
```

```
[
 clrscr();
 gotoxy(10,20);
 highvideo();
 cprintf("File already exists !!!");
 normvideo();
 delay(1000);
 fclose(fp);
 clrscr();
 goto cd;
]
 clrscr();
 flushall();
 switch(c)
 [
  case '1' : [
                                   fp = fopen(name,"w");
                                   gotoxy(5,15);
                                   printf("Kindly give #sources ( = #species) ");
                                   srcl=0;
                                   scanf("%d",&srcl);
                                   flushall();
                                   fprintf(fp,"%d\n",srcl);
                                   clrscr();
                                   gotoxy(10,10);
                                   printf("Name of the next source ? ");
                                   for (i=0;i<srcl;i++)
                                   [
                                    src_name[i] = malloc(15);
                                    *src_name[i] = NULL;
                                    gets(name1);
                                    strcpy(src_name[i],name1);
                                    fprintf(fp,"%s\t",src_name[i]);
                                    flushall();
                                    gotoxy(36,10);
                                    clreol();
                                   ]
                                   fprintf(fp,"\n");
                                   clrscr();
                                   gotoxy(10,10);
                                   printf("Name of the next species ? ");
                                   for (i=0;i<srcl;i++)
                                   [
                                    spc_name[i] = malloc(15);
                                    *spc_name[i] = NULL;
                                    gets(name2);
                                    strcpy(spc_name[i],name2);
                                    fprintf(fp,"%s\t",spc_name[i]);
                                    flushall();
                                    gotoxy(37,10);
                                    clreol();
                                   ]
                                   spc_name[srcl] = malloc(15);
                                   *spc_name[srcl] = NULL;
                                   spc_name[srcl]="concentrations";
                                   fprintf(fp,"concentrations\n");
                                   clrscr();
                                   flushall();
                                   highvideo();
                                   cprintf("\n\n\tKindly input the coefficients");
                                   normvideo();
                                   for (i=0;i<srcl;i++)
                                   [
```

```
gotoxy(1,15);
clreol();
printf("%s",src_name[i]);
for (j=0;j<=srcl;j++)
{
  gotoxy(15,10);
  clreol();
  printf("%s",spc_name[j]);
  gotoxy(15,15);
  clreol();
  coeff=0;
  scanf("%f",&coeff);
  fprintf(fp,"%10.7f\t",coeff);
}
fprintf(fp,"\n");
}
fclose(fp);
for (j=0;j<=srcl;j++)
  free(spc_name[j]);
for (j=0;j<srcl;j++)
  free(src_name[j]);
break;
} /* end of case-1 for methods*/
case '2' : {
fp = fopen(name,"w");
*name = NULL;
gotoxy(5,15);
printf("Kindly give #sources  ");
srcl=0;
scanf("%d",&srcl);
fprintf(fp,"%d\t",srcl);
gotoxy(5,17);
printf("Kindly give #species  ");
spcl=0;
scanf("%d",&spcl);
fprintf(fp,"%d\n",spcl);
clrscr();
gotoxy(10,10);
flushall();
printf("Name of the next source ? ");
for (i=0;i<srcl;i++)
{
  src_name[i] = malloc(15);
  *src_name[i] = NULL;
  gets(src_name[i]);
  fprintf(fp,"%s\t",src_name[i]);
  gotoxy(36,10);
  clreol();
}
fprintf(fp,"\n");
clrscr();
gotoxy(10,10);
printf("Name of the next species ? ");
for (i=0;i<spcl;i++)
{
  spc_name[i] = malloc(15);
  *spc_name[i] = NULL;
  gets(spc_name[i]);
  fprintf(fp,"%s\t",spc_name[i]);
  gotoxy(36,10);
  clreol();
}
fprintf(fp,"\n");
```

```
clrscr();
gotoxy(6,6);
highvideo();
cprintf("R E A D I N G   C O N C E N T R A T I O N S");
normvideo();
gotoxy(3,16);
printf("Input concentrations observed for ");
for (i=0;i<spcl;i++)
{
  gotoxy(37,16);
  clreol();
  printf("%s : ",spc_name[i]);
  conc=0;
  scanf("%f",&conc);
  fprintf(fp,"%10.7f\t",conc);
}
fprintf(fp,"\n");
clrscr();
printf("\n\n\tKindly input the coefficients");
for (i=0;i<srcl;i++)
{
  gotoxy(1,15);
  clreol();
  printf("%s",src_name[i]);
  for (j=0;j<spcl;j++)
  {
    gotoxy(15,10);
    clreol();
    printf("%s",spc_name[j]);
    gotoxy(15,15);
    clreol();
    coeff=0;
    scanf("%f",&coeff);
    fprintf(fp,"%10.7f\t",coeff);
  }
  fprintf(fp,"\n");
}
clrscr();
gotoxy(6,6);
highvideo();
cprintf("R E A D I N G   U N C E R T A I N T I E S");
normvideo();
gotoxy(3,16);
printf("Input uncertainties observed for ");
for (i=0;i<spcl;i++)
{
  gotoxy(36,16);
  clreol();
  printf("%s : ",spc_name[i]);
  uncert=0;
  scanf("%f",&uncert);
  fprintf(fp,"%10.7f\t",uncert);
}
fprintf(fp,"\n");
fclose(fp);
for (j=0;j<srcl;j++)
{
  free(spc_name[j]);
  free(src_name[j]);
}
clrscr();
break;
```

```
                case '3' : {
                                fp = fopen(name,"w");
                                *name = NULL;
                                gotoxy(5,15);
                                printf("Kindly give no. of variables: ");
                                spcl=0;
                                scanf("%d",&spcl);
                                flushall();
                                fprintf(fp,"\t%d\t",spcl);
                                gotoxy(5,17);
                                printf("Kindly give no. of cases: ");
                                srcl=0;
                                scanf("%d",&srcl);
                                flushall();
                                fprintf(fp,"%d\n",srcl);
                                clrscr();
                                gotoxy(10,10);
                                clrscr();
                                for (i=0;i<srcl;i++)
                                {
                                  for (j=0;j<spcl;j++)
                                  {
                                    gotoxy(10,15);
                                    printf("Input value for");
                                    printf(" case %d variable %d (Format:3.6): ",i+1,j+1);
                                    scanf("%f",&x[i][j]);
                                    fprintf(fp,"\t%10.6f",x[i][j]);
                                    gotoxy(22,15);
                                    clreol();
                                  }
                                  fprintf(fp,"\n");
                                }
                                flushall();
                                fclose(fp);
                                clrscr();
                           }
                      }
                  break;
              } /* end of case-1 in 1-7 menu */


case '2': {


              //////////////////////////////
             ///  E D I T   F I L E  ///
             //////////////////////////////
                  sound(700); delay(200);
                  sound(800); delay(200);
                  sound(900); delay(200);
                  nosound();

                  restorecrtmode();
                  clrscr();
                  gotoxy(5,25);
                  highvideo();
                  cprintf("N O T E : ");
                  lowvideo();
                  printf("You can choose only the data-files with extensions BLM/LSQ/FAC used in this software");
                  normvideo();
                  gotoxy(10,10);
```

```
                printf("2.  xxxxxxxx.LSQ");
                gotoxy(10,12);
                printf("3.  xxxxxxxx.FAC");
                gotoxy(10,14);
                highvideo();
                cprintf("Press Y to EXIT else any other key");
                normvideo();
                if ((c=getch()) == 'y' || c == 'Y')
                        goto last;
                gotoxy(10,14);
                clreol();
                gotoxy(10,17);
                printf("KINDLY GIVE THE FILE-NAME  : ");
                gets(fname1);
                clrscr();
                if(strstr(fname1,".elm") != NULL || strstr(fname1,".ELM") != NULL)
                {
hil  :                  fp = fopen(fname1,"r+");
                        if (!fp)
                        {
                          highvideo();
                          cprintf("\n\n\tCan't open the file ...!!!...");
                          delay(1500);
                          normvideo();
                          goto ww;
                        }
                        fgetpos(fp,&filepos);
aa  :                   gotoxy(10,5);
                        normvideo();
                        cprintf("Kindly give the source-name : ");
                        gets(name1);
                        n=0;
                        fscanf(fp,"%d",&n);
                        for (i=0;i<n;i++)
                        {
                          fscanf(fp,"%s",&fname2);
                          if (strcmp(fname2,name1) == 0)
                            goto bb;
                        }
                        highvideo();
                        cprintf("\n\n\n\tNo such source exists ...!!!");
                        normvideo();
                        delay(1000);
                        clrscr();
                        fsetpos(fp,&filepos);
                        goto aa;


bb  :                   fsetpos(fp,&filepos);
                        clrscr();
                        gotoxy(10,5);
                        normvideo();
                        cprintf("Kindly give the species-name : ");
                        gets(name2);
                        n=0;
                        fscanf(fp,"%d",&n);
                        for (i1=0;i1<n;i1++)
                          fscanf(fp,"%s",&path);
                        for (j=0;j<=n;j++)
                        {
                          fscanf(fp,"%s",&fname2);
                          if (strcmp(fname2,name2) == 0)
```

```
    goto cc;
}
highvideo();
cprintf("\n\n\n\tNo such species exists ...!!!");
normvideo();
delay(1000);
clrscr();
goto bb;
fsetpos(fp,&filepos);
fscanf(fp,"%d",&n);
for(i1=0;i1<n;i1++)
  fscanf(fp,"%s",&path);
for(i1=0;i1<=n;i1++)
  fscanf(fp,"%s",&path);
for (i1=0;i1<i;i1++)
{
  for (j1=0;j1<=n;j1++)
  {
    fgetpos(fp,&filepos);
    coeff=0;
    fscanf(fp,"%f",&coeff);
  }
}
for (j1=0;j1<=j;j1++)
{
  fgetpos(fp,&filepos);
  fscanf(fp,"%f",&coeff);
}

clrscr();
gotoxy(15,10);
highvideo();
cprintf("E D I T I N G   F I L E");
normvideo();
gotoxy(20,15);
printf("%s",name2);
gotoxy(2,17);
printf("%s",name1);
gotoxy(22,17);
printf("%10.7f",coeff);
highvideo();
gotoxy(5,24);
cprintf("Kindly input new value : ");
normvideo();
coeff=0;
scanf("%f",&coeff);
fsetpos(fp,&filepos);
if (j==0)
   fprintf(fp,"\n%10.7f\t",coeff);
else
   fprintf(fp,"\t%10.7f\t",coeff);
clrscr();
flushall();
gotoxy(5,14);
highvideo();
cprintf("WANT TO EDIT MORE ");
lowvideo();
cprintf("(Y/N) ");
highvideo();
cprintf("? ");
normvideo();
fcloseall();
c=getch();
```

```
                clrscr();
                if (c=='y' || c=='Y')
                  goto hi1;

          }

          else

          if(strstr(fname1,".lsq") != NULL || strstr(fname1,".LSQ") != NULL)
          {
hi2 :           fp = fopen(fname1,"r+");
                 if (!fp)
                 {
                   highvideo();
                   cprintf("\n\n\tCan't open the file ...!!!...");
                   delay(1500);
                   normvideo();
                   goto ww;
                 }
        fgetpos(fp,&filepos);
                 clrscr();
                 gotoxy(5,10);
                 highvideo();
                 cprintf("You can edit %s for the followings : ",fname1);
                 gotoxy(10,12);
                 cprintf("1.  ");
                 printf("S o u r c e   C o m p o s i t i o n s");
                 gotoxy(10,13);
                 cprintf("2.  ");
                 printf("S p e c i e s   C o n c e n t r a t i o n s");
                 gotoxy(10,14);
                 cprintf("3.  ");
                 printf("U n c e r t a i n t i e s ");
                 gotoxy(10,15);
                 cprintf("4.  ");
                 printf("E X I T   M E N U");
                 gotoxy(5,18);
                 cprintf("E N T E R   C H O I C E [1-4]");
                 normvideo();
                 c = getch();
                 switch(c)
                 {
                   case '1' :{
aa1 :                             clrscr();
                                    gotoxy(10,5);
                                    normvideo();
                                    cprintf("Kindly give the source-name : ");
                                    gets(name1);
                                    fscanf(fp,"%d",&n);
                                    fscanf(fp,"%d",&n1);
                                    for (i=0;i<n;i++)
                                    {
                                      fscanf(fp,"%s",&fname2);
                                      if (strcmp(fname2,name1) == 0)
                                        goto bb1;
                                    }
                                    highvideo();
                                    cprintf("\n\n\n\tNo such source exists ...!!!
                                    normvideo();
                                    delay(1000);
                                    clrscr();
```

```
fsetpos(fp,&filepos);
goto aa1;

fsetpos(fp,&filepos);
clrscr();
gotoxy(10,5);
normvideo();
cprintf("Kindly give the species-name : ");
gets(name2);
fscanf(fp,"%d",&n);
fscanf(fp,"%d",&n1);
for (i1=0;i1<n;i1++)
  fscanf(fp,"%s",&path);
for (j=0;j<n1;j++)
{
  fscanf(fp,"%s",&fname2);
  if (strcmp(fname2,name2) == 0)
    goto cc1;
}
highvideo();
cprintf("\n\n\n\tNo such species exists ...!!!");
normvideo();
delay(1000);
clrscr();
goto bb1;


fsetpos(fp,&filepos);
fscanf(fp,"%d",&n);
fscanf(fp,"%d",&n1);
for(i1=0;i1<n;i1++)
  fscanf(fp,"%s",&path);
for(i1=0;i1<n1;i1++)
  fscanf(fp,"%s",&path);
for (i1=0;i1<n1;i1++)
  fscanf(fp,"%f",&coeff);
for (i1=0;i1<i;i1++)
{
  for (j1=0;j1<n1;j1++)
    {
      fgetpos(fp,&filepos);
      fscanf(fp,"%f",&coeff);
    }
}
  for (j1=0;j1<=j;j1++)
    {
      fgetpos(fp,&filepos);
      fscanf(fp,"%f",&coeff);
    }

  clrscr();
  gotoxy(15,10);
  highvideo();
  cprintf("E D I T I N G   F I L E");
  normvideo();
  gotoxy(20,15);
  printf("%s",name2);
  gotoxy(2,17);
  printf("%s",name1);
  gotoxy(22,17);
  printf("%10.7f",coeff);
  highvideo();
  gotoxy(5,24);
```

```
                    cprintf("Kindly input new value : ");
                    normvideo();
                    coeff=0;
                    scanf("%f",&coeff);
                    fsetpos(fp,&filepos);
                    if (j==0)
                      fprintf(fp,"\n%10.7f\t",coeff);
                    else
                      fprintf(fp,"\t%10.7f\t",coeff);
                    fclose(fp);
                    break;
                } /*end case 1*/

            case '2' :{
bbbl :
                    fsetpos(fp,&filepos);
                    clrscr();
                    gotoxy(10,5);
                    normvideo();
                    cprintf("Kindly give the species-name : ");
                    gets(name2);
                    fscanf(fp,"%d",&n);
                    fscanf(fp,"%d",&nl);
                    for (il=0;il<n;il++)
                      fscanf(fp,"%s",&path);
                    for (j=0;j<nl;j++)
                    {
                      fscanf(fp,"%s",&fname2);
                      if (strcmp(fname2,name2) == 0)
                        goto cccl;
                    }
                    highvideo();
                    cprintf("\n\n\tNo such species exists ...!!!");
                    normvideo();
                    delay(1000);
                    clrscr();
                    goto bbbl;

cccl :
                    fsetpos(fp,&filepos);
                    fscanf(fp,"%d",&n);
                    fscanf(fp,"%d",&nl);
                    for (il=0;il<n;il++)
                      fscanf(fp,"%s",&path);
                    for (il=0;il<nl;il++)
                      fscanf(fp,"%s",&path);
                    for (il=0;il<=j;il++)
                      {
                        fgetpos(fp,&filepos);
                        fscanf(fp,"%f",&coeff);
                      }
                    clrscr();
                    gotoxy(15,10);
                    highvideo();
                    cprintf("E D I T I N G   F I L E");
                    normvideo();
                    gotoxy(1,15);
                    printf("Concentration for %s : %10.7f",name2,coeff);
                    highvideo();
                    gotoxy(5,20);
                    cprintf("Kindly input new value : ");
                    normvideo();
                    coeff=0;
                    scanf("%f",&coeff);
                    fsetpos(fp,&filepos);
```

```
                    if (j==0)
                      fprintf(fp,"\n%10.7f\t",coeff);
                    else
                      fprintf(fp,"\t%10.7f\t",coeff);
                    fclose(fp);
                    break;
                } /* end of case 2 */

case '3':{
                    fsetpos(fp,&filepos);
                    clrscr();
                    gotoxy(10,5);
                    normvideo();
                    cprintf("Kindly give the species-name : ");
                    gets(name2);
                    fscanf(fp,"%d",&n);
                    fscanf(fp,"%d",&n1);
                    for (i1=0;i1<n;i1++)
                      fscanf(fp,"%s",&path);
                    for (j=0;j<n1;j++)
                    {
                      fscanf(fp,"%s",&fname2);
                      if (strcmp(fname2,name2) == 0)
                        goto clc1;
                    }
                    highvideo();
                    cprintf("\n\n\n\tNo such species exists ...!!!");
                    normvideo();
                    delay(1000);
                    clrscr();
                    goto blbl;

                    fsetpos(fp,&filepos);
                    fscanf(fp,"%d",&n);
                    fscanf(fp,"%d",&n1);
                    for (i1=0;i1<n;i1++)
                      fscanf(fp,"%s",&path);
                    for (i1=0;i1<n1;i1++)
                      fscanf(fp,"%s",&path);
                    for (i1=0;i1<n1;i1++)
                      fscanf(fp,"%f",&coeff);
                    for (i1=0;i1<n;i1++)
                      for (j1=0;j1<n1;j1++)
                         fscanf(fp,"%f",&coeff);
                    for (i1=0;i1<=j;i1++)
                      {
                        fgetpos(fp,&filepos);
                        fscanf(fp,"%f",&coeff);
                      }
                    clrscr();
                    gotoxy(15,10);
                    highvideo();
                    cprintf("E D I T I N G   F I L E");
                    normvideo();
                    gotoxy(1,15);
                    printf("Uncertainty for %s : %10.7f",name2,coeff);
                    highvideo();
                    gotoxy(5,20);
                    cprintf("Kindly input new value : ");
                    normvideo();
                    coeff=0;
                    scanf("%f",&coeff);
                    fsetpos(fp,&filepos);
```

```
                                        if (j==0)
                                          fprintf(fp,"\n%10.7f",coeff);
                                        else
                                          fprintf(fp,"\t%10.7f\t",coeff);
                                        fclose(fp);
                                        break;
                                    }
                    case '4':           goto last;
                    default :{
                                        sound(200); delay(100);
                                        sound(300); delay(100);
                                        nosound();
                                    }
                   }  /* end of switch */
                   clrscr();
                   flushall();
                   gotoxy(5,14);
                   highvideo();
                   cprintf("WANT TO EDIT MORE ");
                   lowvideo();
                   cprintf("(Y/N) ");
                   highvideo();
                   cprintf("? ");
                   normvideo();
                   fcloseall();
                   c=getch();
                   clrscr();
                   if (c=='y' || c=='Y')
                      goto hi2;

          }
          else
          if(strstr(fname1,".fac") != NULL || strstr(fname1,".FAC") != NULL)
          {
                 clrscr();
                 gotoxy(10,5);
                 highvideo();
                 printf("E D I T I N G  F A C T O R  D A T A - F I L E");
normvideo();
                 fp = fopen(fname1,"r+");
                   if (!fp)
                   {
                     gotoxy(10,10);
                     highvideo();
                     cprintf("\n\n\tCan't open the file ...!!!...");
                     delay(1500);
                     normvideo();
                     goto ww;
                   }
                 fgetpos(fp,&filepos);
                 gotoxy(7,15);
                 printf("The variable no. ? ");
                 scanf("%d",&n1);
                 flushall();
                 gotoxy(7,20);
                 printf("The case no. ? ");
                 scanf("%d",&n);
                 flushall();
                 fscanf(fp,"%d",&i1);
                 fscanf(fp,"%d",&i2);
                 fgetpos(fp,&filepos);
                 if (n1 > i1 || n > i2)
                 {
```

```
          gotoxy(10,22);
          highvideo();
          printf("This entry doesn't exist...!!!");
          normvideo();
          delay(1000);
          gotoxy(10,22);
          clreol();
          gotoxy(7,20);
          clreol();
          gotoxy(7,15);
          clreol();
          goto tau;
        }
        for (i=0;i<n-1;i++)
          for (j=0;j<il;j++)
          {
            fgetpos(fp,&filepos);
            fscanf(fp,"%f",&coeff);
          }
        for (j=0;j<nl;j++)
          {
            fgetpos(fp,&filepos);
            fscanf(fp,"%f",&coeff);
          }
        flushall();
        gotoxy(7,22);
        printf("The current value is (Format:3.6): %10.6f",coeff);
        fsetpos(fp,&filepos);
        gotoxy(7,24);
        highvideo();
        printf("Enter New Value (Format:3.6): ");
        normvideo();
        scanf("%f",&conc);
        if(nl==1)
          fprintf(fp,"\n\t%10.6f\t",conc);
        else
          if (nl==il)
            fprintf(fp,"\t%10.6f\n\t",conc);
          else
            fprintf(fp,"\t%10.6f\t",conc);
        fclose(fp);
        clrscr();
        flushall();
        gotoxy(5,14);
        highvideo();
        cprintf("WANT TO EDIT MORE ");
        lowvideo();
        cprintf("(Y/N) ");
        highvideo();
        cprintf("? ");
        normvideo();
        fcloseall();
        c=getch();
        clrscr();
        if (c=='y' || c=='Y')
          goto si;
    }
  break;
  }

case '3': {
```

```
 //////////////////////////////
 ///   C O P Y  F I L E   ///
//////////////////////////////


        sound(700); delay(200);
        sound(800); delay(200);
        sound(900); delay(200);
        nosound();
        restorecrtmode();
        clrscr();
        gotoxy(10,5);
        printf("N O T E : You can use WILDCARDS too !!!");
        *name=NULL;
        strcat(name,"COPY ");
        gotoxy(1,10);
        printf("Name of the file to be COPIed : ");
        gets(fname1);
        strcat(name,fname1);
        strcat(name," ");
        gotoxy(1,15);
        printf("\nName of the file to be copied to :");
        gets(fname2);
        strcat(name,fname2);
        system(name);
        printf("\n\nPRESS ANY KEY !!!");
        getch();
        clrscr();
        break;
      }


case  '4': {

 //////////////////////////////
 /// R E N A M E  F I L E  ///
//////////////////////////////


        sound(700); delay(200);
        sound(800); delay(200);
        sound(900); delay(200);
        nosound();
        restorecrtmode();
        clrscr();
        gotoxy(10,5);
        printf("N O T E : You can use WILDCARDS too !!!");
        *name=NULL;
        strcat(name,"RENAME ");
        gotoxy(1,10);
        printf("Name fo the file to be RENAMEd : ");
        gets(fname1);
        strcat(name,fname1);
        strcat(name," ");
        gotoxy(1,15);
        printf("\nNew name of the file : ");
        gets(fname2);
        strcat(name,fname2);
        system(name);
        printf("\n\nPRESS ANY KEY !!!");
        getch();
        clrscr();
        break;
      [redacted]
```

```
                }

case  '5': {

            ///////////////////////////
           /// D E L E T E  F I L E ///
          ////////////////////////////


                sound(700); delay(200);
                sound(800); delay(200);
                sound(900); delay(200);
                nosound();
                restorecrtmode();
                clrscr();
                gotoxy(10,5);
                printf("N O T E : You can use WILDCARDS too !!!");
                *name=NULL;
                strcat(name,"DEL ");
                gotoxy(1,10);
                printf("Name of the file to be DELETEd : ");
                gets(fname1);
                strcat(name,fname1);
                system(name);
                printf("\n\nPRESS ANY KEY !!!");
                getch();
                break;
             }


case  '6': {

            ////////////////////////////
           /// L I S T   F I L E S   ///
          ////////////////////////////



                sound(700); delay(200);
                sound(800); delay(200);
                sound(900); delay(200);
                nosound();
                restorecrtmode();
                clrscr();
                gotoxy(10,5);
                printf("N O T E : You can use WILDCARDS too !!!");
                gotoxy(1,10);
                cprintf("Please give path for DIR  : ");
                gets(path);
                *path1=NULL;
                strcat(path1,"dir/p ");
                strcat(path1, path);
                system(path1);
                printf("\n\nPRESS ANY KEY !!!");
                getch();
                break;
             }
case  '7': {
               sound(700); delay(200);
               sound(800); delay(200);
               sound(900); delay(200);
               nosound();
```

```
                              closegraph(); exit(c);
                            }
          default  : {
                              sound(500); delay(150);
                              sound(1000); delay(150);
                              nosound();
                              goto b;
                            }
    }
last  :  fcloseall();
          flushall();
          goto a;
}
```

/********** M E N U 4 . C ************/

/************ M E N U 4 ***************/

```
#include <graphics.h>
#include <conio.h>
int main()
{
            int gm,gd=DETECT,j1,j2,c;
            int poly[8] = {275,55,285,50,285,60,275,55};




            j1 = getcolor();
            j2 = getbkcolor();
            initgraph(&gd,&gm,"");
            rectangle(150,145,220,230);
            outtextxy(173,150,"A");
            outtextxy(173,160,"N");
            outtextxy(173,170,"A");
            outtextxy(173,180,"L");
            outtextxy(173,190,"Y");
            outtextxy(173,200,"S");
            outtextxy(173,210,"I");
            outtextxy(173,220,"S");
            sound(750); delay(200); nosound(); delay(200);
            rectangle(550,20,630,50);
            outtextxy(560,30,"C.M.B.");
            circle(650,35,10);
            outtextxy(648,33,"1");
            line(200,185,550,35);
            line(200,183,550,33);
            sound(750); delay(200); nosound(); delay(200);
            rectangle(550,170,630,200);
            outtextxy(560,175,"FACTOR");
            outtextxy(560,185,"ANALYSIS");
            circle(650,185,10);
            outtextxy(648,183,"2");
            line(200,185,550,185);
            line(200,183,550,183);
            sound(750); delay(200); nosound(); delay(200);
            rectangle(550,315,630,345);
            outtextxy(560,325,"EXIT");
            circle(650,330,10);
            outtextxy(648,328,"3");
            line(200,185,550,335);
            line(200,183,550,333);
            sound(750); delay(200); nosound(); delay(200);
            circle(260,55,10);
            drawpoly(4,poly);
            rectangle(285,53,295,58);
            floodfill(1,1,WHITE);
            setcolor(j2);
            outtextxy(50,50,"E N T E R   C H O I C E ");
            sound(500); delay(200); nosound(); delay(200);
            sound(500); delay(200); nosound(); delay(200);
            sound(500); delay(200); nosound(); delay(200);
            c = getch();
            switch(c)
```

```
{
  case '1' :
  case '2' :
  case '3' : { closegraph(); exit(c);}
  default  : {sound(250); delay(150); sound(1000); delay(150); nosound(); goto b;}
}
closegraph();
```

```
/********** M E N U 5 . C ************/

                    /************ M E N U 5 ***************/

#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int gm,gd=DETECT,j1,j2;


int main()
{

        int i,c,temp1,temp2;
        void *buffer;
        unsigned int size;
        int sbrt1();
        int sbrt2();



        j1 = getcolor();
        j2 = getbkcolor();
        initgraph(&gd,&gm,"");
        rectangle(100,100,600,200);
        floodfill(101,101,WHITE);
        setcolor(j2);
        outtextxy(150,150,"D O   Y O U   W A N T   TO  C O N T I N U E  (Y/N)");
        rectangle(555,143,575,160);
        size = imagesize(555,143,575,160);
        buffer = (void *)malloc(size);
        getimage(555,143,575,160,(void *)buffer);
        putimage(555,143,(void *)buffer,XOR_PUT);
        free(buffer);
a  :    c = getch();
        switch(c)
        {
          case 'Y':
          case 'y': {
                      temp1 = sbrt1();
                      temp2 = sbrt2();
                      closegraph();
                      if (temp1=='1')
                          {
                            if (temp2=='1')
                              exit(1);
                            else
                                 if (temp2=='2')
                                        exit(2);
                                 else
                                        exit(3);
                          }
                      else
                          if (temp1=='2')
                            {
                              if (temp2=='1')
                                exit(4);
                              else
                                if (temp2=='2')
```

```
                                        exit(5);
                                else
                                        exit(6);
                            }
                        }
            case 'N':
            case 'n': { closegraph(); exit(c); }
            default : {
                        sound(500); delay(100);
                        sound(700); delay(100);
                        nosound();
                        goto a;
                        }
        }
}


int sbrt1()
{
   int c,i;
   void *buffer;
   unsigned int size,width;
   char name[8];


   cleardevice();
   initgraph(&gd,&gm,"");
   bar(150,150,600,175);
   rectangle(150,175,600,200);
   outtextxy(180,185," 1.   Screen                    2.   File");
   setcolor(j2);
   outtextxy(180,160,"K I N D L Y   G I V E   ‡D A T A - S O U R C E");
   rectangle(570,157,590,170);
   size = imagesize(570,157,590,170);
   buffer = (void *)malloc(size);
   getimage(570,157,590,170,(void *)buffer);
   putimage(570,157,(void *)buffer,XOR_PUT);
   free(buffer);
a :c = getch();
   switch(c)
   {
     case '1'  :
     case '2'  : {
                        return(c);
                      }
     default : {
                      sound(700); delay(100);
                      sound(900); delay(100);
                      nosound();
                      goto a;
                 }
   }
}



int sbrt2()
{
   int c,i;
   void *buffer;
   unsigned int size,width;
   char name[8];
```

```
   cleardevice();
   initgraph(&gd,&gm,"");
   bar(150,150,600,175);
   rectangle(150,175,600,200);
   outtextxy(180,185," 1.   Screen      2.   Printer       3.   File");
   setcolor(j2);
   outtextxy(180,160,"K I N D L Y  G I V E  ‡ D E S T I N A T I O N");
   rectangle(570,157,590,170);
   size = imagesize(570,157,590,170);
   buffer = (void *)malloc(size);
   getimage(570,157,590,170,(void *)buffer);
   putimage(570,157,(void *)buffer,XOR_PUT);
   free(buffer);
a :c = getch();
   switch(c)
   {
     case '1'  :
     case '2'  :
     case '3'  : {
                            return(c);
                          }
     default   : {
                            sound(700); delay(100);
                            sound(900); delay(100);
                            nosound();
                            goto a;
                          }
   }
 }
```

/********** M E N U 6 . C ************/

/************ M E N U 6 ***************/

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int gm,gd=DETECT,j1,j2;


int main()
{
    void *buffer, *buffer1;
    unsigned int size,size1;
    int i,tempx1,tempx2,j,c;



    initgraph(&gd,&gm,"");
    j1 = getcolor();
    j2 = getbkcolor();
    bar(225,50,400,100);
    delay(500);
    size = imagesize(225,60,250,90);
    getimage(225,60,230,90,(void *)buffer);
    j=0;
    for (i=0;i<145;i++)
    {
      putimage((tempx1=220-i),60,(void *)buffer,COPY_PUT);
      putimage(400+i,60,(void *)buffer,COPY_PUT);
      sound(50+i);
      delay(50);
      j = j+5;
    }
    nosound();
    tempx2=401;
    free(buffer);
    setcolor(j2);
    outtextxy(230,75,"CHEMICAL MASS BALANCE");
    size = imagesize(tempx1,60,224,60);
    buffer = (void *)malloc(size);
    for (i=60;i<200;i++)
    {
      getimage(tempx1,i+29,224,i+29,(void *)buffer);
      putimage(tempx1,i,(void *)buffer,XOR_PUT);
      putimage(tempx2,i,(void *)buffer,XOR_PUT);
      putimage(tempx1,i+31,(void *)buffer,COPY_PUT);
      putimage(tempx2,i+31,(void *)buffer,COPY_PUT);
      sound(200-i);
      delay(50);
    }
    nosound();
    free(buffer);
    outtextxy(tempx1+3,210,"MATRIX-ELIMINATION");
    outtextxy(tempx2+20,210,"LEAST-SQUARE");
    setcolor(j1);
    line(224,75,160,75);
    line(401,75,465,75);
    line(160,75,160,199);
```

```
    line(465,75,465,199);
    circle(145,185,10);
    circle(480,185,10);
    outtextxy(143,183,"1");
    outtextxy(477,183,"2");
    rectangle(445,285,470,300);
    floodfill(446,286,WHITE);
    outtextxy(160,250,"You can select '3' if you want to exit this menu");
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
    outtextxy(150,280,"Please Enter Choice [1-3]");
a : c = getch();
 switch(c)
    {
      case '1'  :
      case '2'  :
      case '3'  : { closegraph(); exit(c); }
      default   : {
                              sound(2000); delay(100);
                              sound(1000); delay(100);
                              nosound();
                              goto a;
                            }
    }
    closegraph();
 }
```

/********** M E N U 7 . C ************/

/************ M E N U 7 ****************/

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int gm,gd=DETECT,j1,j2;


int main()
{
    void *buffer, *buffer1;
    unsigned int size,size1;
    int i,tempx1,tempx2,j,c;



    initgraph(&gd,&gm,"");
    j1 = getcolor();
    j2 = getbkcolor();
    bar(225,50,400,100);
    delay(500);
    size = imagesize(225,60,250,90);
    getimage(225,60,230,90,(void *)buffer);
    j=0;
    for (i=0;i<145;i++)
    {
      putimage((tempx1=220-i),60,(void *)buffer,COPY_PUT);
      putimage(400+i,60,(void *)buffer,COPY_PUT);
      sound(50+i);
      delay(50);
      j = j+5;
    }
    nosound();
    tempx2=401;
    free(buffer);
    setcolor(j2);
    outtextxy(245,75,"FACTOR ANALYSIS");
    size = imagesize(tempx1,60,224,60);
    buffer = (void *)malloc(size);
    for (i=60;i<200;i++)
    {
      getimage(tempx1,i+29,224,i+29,(void *)buffer);
      putimage(tempx1,i,(void *)buffer,XOR_PUT);
      putimage(tempx2,i,(void *)buffer,XOR_PUT);
      putimage(tempx1,i+31,(void *)buffer,COPY_PUT);
      putimage(tempx2,i+31,(void *)buffer,COPY_PUT);
      sound(200-i);
      delay(50);
    }
    nosound();
    free(buffer);
    outtextxy(tempx1+7,210,"ROTATION-1");
    outtextxy(tempx2+20,210,"ROTATION-2");
    setcolor(j1);
    line(224,75,160,75);
    line(401,75,465,75);
    line(160,75,160,199);
```

```
   line(465,75,465,199);
   circle(145,185,10);
   circle(480,185,10);
   outtextxy(143,183,"1");
   outtextxy(477,183,"2");
   rectangle(445,285,470,300);
   floodfill(446,286,WHITE);
   outtextxy(160,250,"You can select '3' if you want to exit this menu");
   settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
   outtextxy(150,280,"Please Enter Choice [1-3]");
a : c = getch();
 switch(c)
    {
      case '1'  :
      case '2'  :
      case '3'  : { closegraph(); exit(c); }
      default   : {
                              sound(2000); delay(100);
                              sound(1000); delay(100);
                              nosound();
                              goto a;
                            }
    }
    closegraph();
 }
```

/* QMAX1.C */

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "qrtmax.c"
#define MAX 50


int main()
{
  int  nl=0,nsp,nr,i,j,il,jl;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],rl[MAX][MAX],var[MAX],d[MAX][MAX],
          b[MAX][MAX],fact[MAX][MAX],sum,suml;



  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F A C T O R   A N A L Y S I S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("Q U A R T I M A X - R O T A T I O N");
  normvideo();
  delay(2000);
  clrscr();


  gotoxy(10,5);
  printf("No. of variables ? : ");
  scanf("%d",&nsp);
  flushall();
  gotoxy(10,10);
  printf("No. of cases ? : ");
  scanf("%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
    for (j=0;j<nsp;j++)
    {
      gotoxy(10,15);
      printf("Input value for case %d variable %d : ",i+1,j+1);
      scanf("%f",&x[i][j]);
      gotoxy(22,15);
      clreol();
    }
 flushall();
 clrscr();

 gotoxy(10,5);
 printf("Press any key and you see the raw data...!!!");
 getch();
 flushall();
 printf("\n\n");

for (i=0;i<nr;i++)
{
```

```
  for (j=0;j<nsp;j++)
   printf("%f  ",x[i][j]);
  printf("\n");
 }
printf("\n\n");
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();


correl(x,nsp,nr,r);
gotoxy(10,5);
printf("!!!...Press any key again to see the correlation matrix...!!!");
getch();
flushall();
printf("\n\n");

for (i=0;i<nsp;i++)
{
  for (j=0;j<nsp;j++)
   printf("%f  ",r[i][j]);
  printf("\n");
}
printf("\n\n");
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();

if (mat_inv(r,nsp,r1)==0) goto out;

printf("/...Press a key to see the communalities found by SMC method...\\");
getch();
flushall();
printf("\n\n");

sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  printf("\n\t%f\n",commun[i]);
}
printf("\n\n");
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();
gotoxy(10,15);
highvideo();
cprintf("PLEASE WAIT....EIGENVALUES ARE COMING NEXT...");
normvideo();
printf("\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];


nl=factor(r,nsp,d,fact,var);
if (nl < 0) goto out;
delay(2000);
```

```
fcloseall();
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();

gotoxy(10,10);
printf("->>>Press a key and get the Eigenvalues ->>>\n\n");
getch();
flushall();
for (i=0;i<nsp;i++)
  printf("\t%f\n",d[i][i]);
printf("\nObviously you have to PRESS a key to get...");
getch();
flushall();
clrscr();



printf("FACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    printf("%f  ",fact[i][j]);
  printf("\n");
}
printf("\n\n");



printf("VARIANCE: ");
for (j=0;j<nl;j++)
  printf("%f  ",var[j]);
printf("\n\n");

suml = 0.0;
printf("VARIANCE % : ");
for (j=0;j<nl;j++)
{
  suml=suml+(var[j]/sum)*100;
  printf("%6.2f  ",(var[j]/sum)*100);
}
printf("\n\nTotal variance accounted for : %6.2f\n\n",suml);


printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();

gotoxy(5,10);
if(quartimax(fact,nsp,nl,b) < 0)goto out;
flushall();

printf("\n\nPress a key to get the final solution using QUARTIMAX method\n\n");
getch();
flushall();
clrscr();

printf("The final solution after QUARTIMAX Rotation :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
```

```
      printf("%f  ",b[i][j]);
    printf("\n");
  }
  printf("\n\n");

  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();
  clrscr();

  exit(1);
out: printf("\n\nERROR...!!!");
  return(0);
}
```

```
/* QMAX2.C */

#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "qrtmax.c"
#define MAX 50


int main()
{
  int   n1=0,nsp,nr,i,j,i1,j1;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],r1[MAX][MAX],var[MAX],
            fact[MAX][MAX],d[MAX][MAX],b[MAX][MAX],sum,sum1;



  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F  A  C  T  O  R    A  N  A  L  Y  S  I  S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("Q U A R T I M A X - R O T A T I O N");
  normvideo();
  delay(2000);
  clrscr();


  gotoxy(10,5);
  printf("No. of variables ? : ");
  scanf("%d",&nsp);
  flushall();
  gotoxy(10,10);
  printf("No. of cases ? : ");
  scanf("%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
    for (j=0;j<nsp;j++)
    {
      gotoxy(10,15);
      printf("Input value for case %d variable %d : ",i+1,j+1);
      scanf("%f",&x[i][j]);
      gotoxy(22,15);
      clreol();
    }
  flushall();
  fcloseall();
  clrscr();
```

```
fprintf(stdprn,"\n\n\tThe give raw data....\n\n");
for (i=0;i<nr;i++)
{
  for (j=0;j<nsp;j++)
   fprintf(stdprn,"%f  ",x[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");
clrscr();

correl(x,nsp,nr,r);
gotoxy(10,5);
fprintf(stdprn,"The correlation matrix is :\n\n");

for (i=0;i<nsp;i++)
{
  for (j=0;j<nsp;j++)
   fprintf(stdprn,"%f  ",r[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");

if (mat_inv(r,nsp,r1)==0) goto out;

fprintf(stdprn,"The communalities using SMC method are ....\n\n"

sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  fprintf(stdprn,"\n\t%f\n",commun[i]);
}
fprintf(stdprn,"\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];



nl=factor(r,nsp,d,fact,var);
if (nl < 0) goto out;
delay(2000);
clrscr();

gotoxy(10,10);
fprintf(stdprn,"\nThe Eigenvalues are :\n\n");
for (i=0;i<nsp;i++)
  fprintf(stdprn,"\t%f\n",d[i][i]);


fprintf(stdprn,"FACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(stdprn,"%f  ",fact[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");



fprintf(stdprn,"VARIANCE: ");
```

```
   for (j=0;j<nl;j++)
     fprintf(stdprn,"%f  ",var[j]);
   fprintf(stdprn,"\n\n");

   suml = 0.0;
   fprintf(stdprn,"VARIANCE % : ");
   for (j=0;j<nl;j++)
   {
    suml=suml+(var[j]/sum)*100;
    fprintf(stdprn,"%6.2f  ",(var[j]/sum)*100);
   }
   fprintf(stdprn,"\n\nTotal variance accounted for : %6.2f\n\n",suml);

   gotoxy(5,10);
   if(quartimax(fact,nsp,nl,b) < 0)goto out;
   flushall();


   fprintf(stdprn,"\n\nThe final solution after QUARTIMAX Rotation :\n\n");
   for (i=0;i<nsp;i++)
   {
     for (j=0;j<nl;j++)
       fprintf(stdprn,"%f  ",b[i][j]);
     fprintf(stdprn,"\n");
   }
   fprintf(stdprn,"\n\n");


   exit(1);

out: fprintf(stdprn,"\n\nERROR...!!!");
  return(0);
}
```

```
                /* QMAX3.C */


#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "qrtmax.c"
#define  MAX  50


int main()
{
  int   nl=0,nsp,nr,i,j,il,jl;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],rl[MAX][MAX],var[MAX],
            fact[MAX][MAX],b[MAX][MAX],d[MAX][MAX],sum,suml;

  char      c,name[12];

  FILE      *fpl;


  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F  A  C  T  O  R     A  N  A  L  Y  S  I  S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("Q U A R T I M A X - R O T A T I O N");
  normvideo();
  delay(2000);

  clrscr();
aa:  gotoxy(10,10);
  printf("Name of the output-file : ");
  gets(name);
  flushall();


  fpl = fopen(name,"r");
  if (fpl)
  {
    fclose(fpl);
    gotoxy(10,15);
    highvideo();
    cprintf("File already exists...!!!");
    normvideo();
    gotoxy(10,18);
    printf("Overwrite it...(Y/N)...??? ");
    c = getch();
    if (c == 'y' || c == 'Y')
      goto al;
    gotoxy(10,18);
    clreol();
    gotoxy(10,15);
    clreol();
```

```
    gotoxy(25,10);
    clreol();
    goto aa;
  }

al: clrscr();
  fpl = fopen(name,"w");
  if (!fpl)
  {
    fclose(fpl);
    gotoxy(10,15);
    highvideo();
    cprintf("Can't open the file...!!!");
    normvideo();
    delay(300);
    gotoxy(10,15);
    clreol();
    goto al;
  }
  gotoxy(10,5);
  printf("No. of variables ? : ");
  scanf("%d",&nsp);
  flushall();
  gotoxy(10,10);
  printf("No. of cases ? : ");
  scanf("%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
    for (j=0;j<nsp;j++)
    {
      gotoxy(10,15);
      printf("Input value for case %d variable %d : ",i+1,j+1);
      scanf("%f",&x[i][j]);
      gotoxy(22,15);
      clreol();
    }
  flushall();
  clrscr();

  gotoxy(10,5);
  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();


  fprintf(fpl,"\n\n\tThe give raw data....\n\n");
  for (i=0;i<nr;i++)
  {
    for (j=0;j<nsp;j++)
     fprintf(fpl,"%f  ",x[i][j]);
    fprintf(fpl,"\n");
  }
  fprintf(fpl,"\n\n");


 correl(x,nsp,nr,r);
 gotoxy(10,5);
 flushall();
 fprintf(fpl,"The correlation matrix is :\n\n");

 for (i=0;i<nsp;i++)
```

```
{
 for (j=0;j<nsp;j++)
  fprintf(fp1,"%f  ",r[i][j]);
 fprintf(fp1,"\n");
}
fprintf(fp1,"\n\n");

if (mat_inv(r,nsp,r1)==0) goto out;

fprintf(fp1,"The communalities using SMC method are ....\n\n");

sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  fprintf(fp1,"\n\t%f\n",commun[i]);
}
fprintf(fp1,"\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];



nl=factor(r,nsp,d,fact,var);
if (nl < 0) goto out;
delay(2000);
clrscr();

gotoxy(10,10);
fprintf(fp1,"\nThe Eigenvalues are :\n\n");
for (i=0;i<nsp;i++)
  fprintf(fp1,"\t%f\n",d[i][i]);


fprintf(fp1,"FACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(fp1,"%f  ",fact[i][j]);
  fprintf(fp1,"\n");
}
fprintf(fp1,"\n\n");



fprintf(fp1,"VARIANCE: ");
for (j=0;j<nl;j++)
  fprintf(fp1,"%f  ",var[j]);
fprintf(fp1,"\n\n");

sum1 = 0.0;
fprintf(fp1,"VARIANCE % : ");
for (j=0;j<nl;j++)
{
  sum1=sum1+(var[j]/sum)*100;
  fprintf(fp1,"%6.2f  ",(var[j]/sum)*100);
}
fprintf(fp1,"\n\nTotal variance accounted for : %6.2f\n\n",sum1);
gotoxy(5,10);
if(quartimax(fact,nsp,nl,b) < 0)goto out;
flushall();
```

```
  clrscr();

  fprintf(fp1,"The final solution after QUARTIMAX Rotation :\n\n");
  for (i=0;i<nsp;i++)
  {
    for (j=0;j<nl;j++)
      fprintf(fp1,"%f  ",b[i][j]);
    fprintf(fp1,"\n");
  }
  fprintf(fp1,"\n\n");



  exit(1);

out: fprintf(fp1,"\n\nERROR...!!!");
  return(0);
}
```

```
                /*  QMAX4.C  */


#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "qrtmax.c"
#define  MAX  50

int main()
{
  int   n1=0,nsp,nr,i,j,i1,j1;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],r1[MAX][MAX],var[MAX],
            b[MAX][MAX],fact[MAX][MAX],d[MAX][MAX],sum,sum1;

  char  name[12];

  FILE  *fp1;

  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F  A  C  T  O  R    A  N  A  L  Y  S  I  S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("Q U A R T I M A X - R O T A T I O N");
  normvideo();
  delay(2000);

  clrscr();
aa:  gotoxy(10,10);
  printf("Name of the data-file : ");
  gets(name);
  flushall();


  fp1 = fopen(name,"r");
  if (!fp1)
  {
    gotoxy(10,15);
    highvideo();
    cprintf("No such file exists...!!!");
    normvideo();
    delay(500);
    gotoxy(10,15);
    clreol();
    gotoxy(25,10);
    clreol();
    goto aa;
  }


  fscanf(fp1,"%d",&nsp);
  flushall();
  fscanf(fp1,"%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
```

```
    for (j=0;j<nsp;j++)
    {
      fscanf(fp1,"%f",&x[i][j]);
    }
  flushall();
  fcloseall();
  clrscr();

  gotoxy(10,5);
  printf("Press any key and you see the raw data...!!!");
  getch();
  flushall();
  printf("\n\n");

  for (i=0;i<nr;i++)
  {
    for (j=0;j<nsp;j++)
     printf("%f  ",x[i][j]);
    printf("\n");
  }
  printf("\n\n");
  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();
  clrscr();


  correl(x,nsp,nr,r);
  flushall();
  gotoxy(10,5);
  printf("!!!...Press any key again to see the correlation matrix...!!!");
  getch();
  flushall();
  printf("\n\n");

  for (i=0;i<nsp;i++)
  {
    for (j=0;j<nsp;j++)
     printf("%f  ",r[i][j]);
    printf("\n");
  }
  printf("\n\n");
  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();
  clrscr();

  if (mat_inv(r,nsp,r1)==0) goto out;

  printf("/...Press a key to see the communalities found by SMC method...\\");
  getch();
  flushall();
  printf("\n\n");

  sum = 0.0;
  for (i=0;i<nsp;i++)
  {
    commun[i] = 1.0 - (1.0/r1[i][i]);
    sum = sum + commun[i];
    printf("\n\t%f\n",commun[i]);
  }
   printf("\n\n");
  printf("PRESS ANY KEY...!!!");
```

```
getch();
flushall();
clrscr();
gotoxy(10,15);
highvideo();
cprintf("PLEASE WAIT....EIGENVALUES ARE COMING NEXT...");
normvideo();
printf("\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];


nl=factor(r,nsp,d,fact,var);
if (nl < 0) goto out;
delay(2000);
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();

gotoxy(10,10);
printf("->>>Press a key and get the Eigenvalues ->>>\n\n");
getch();
flushall();
for (i=0;i<nsp;i++)
  printf("\t%f\n",d[i][i]);
printf("\nObviously you have to PRESS a key to get...");
getch();
flushall();
clrscr();



printf("FACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    printf("%f  ",fact[i][j]);
  printf("\n");
}
printf("\n\n");


printf("VARIANCE: ");
for (j=0;j<nl;j++)
  printf("%f  ",var[j]);
printf("\n\n");

suml = 0.0;
printf("VARIANCE % : ");
for (j=0;j<nl;j++)
{
  suml=suml+(var[j]/sum)*100;
  printf("%6.2f  ",(var[j]/sum)*100);
}
printf("\n\nTotal variance accounted for : %6.2f\n\n",suml);


printf("PRESS ANY KEY...!!!");
getch();
flushall();
```

```
  clrscr();


  gotoxy(5,10);
  if(quartimax(fact,nsp,nl,b) < 0)goto out;
  flushall();

  printf("\n\nPress a key to get the final solution using QUARTIMAX method\n\n");
  getch();
  flushall();
  clrscr();

  printf("The final solution after QUARTIMAX Rotation :\n\n");
  for (i=0;i<nsp;i++)
  {
    for (j=0;j<nl;j++)
      printf("%f  ",b[i][j]);
    printf("\n");
  }
  printf("\n\n");
  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();
  clrscr();


  exit(1);

out: printf("\n\nERROR...!!!");
  return(0);
}
```

```
/* QMAX5.C */

#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "qrtmax.c"
#define MAX 50

int main()
{
  int   nl=0,nsp,nr,i,j,il,jl;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],rl[MAX][MAX],var[MAX],
             fact[MAX][MAX],b[MAX][MAX],d[MAX][MAX],sum,suml;

  char  name[12];

  FILE  *fpl;

  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F A C T O R   A N A L Y S I S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("Q U A R T I M A X - R O T A T I O N");
  normvideo();
  delay(2000);

  clrscr();
aa:  gotoxy(10,10);
  printf("Name of the data-file : ");
  gets(name);
  flushall();


  fpl = fopen(name,"r");
  if (!fpl)
  {
    gotoxy(10,15);
    highvideo();
    cprintf("No such file exists...!!!");
    normvideo();
    delay(500);
    gotoxy(10,15);
    clreol();
    gotoxy(25,10);
    clreol();
    goto aa;
  }


  fscanf(fpl,"%d",&nsp);
  flushall();
  fscanf(fpl,"%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
```

```
    for (j=0;j<nsp;j++)
    {
      fscanf(fp1,"%f",&x[i][j]);
    }
  flushall();
  clrscr();

  gotoxy(10,5);
  printf("PRESS ANY KEY WHEN THE PRINTER IS READY...!!!");
  getch();
  flushall();

  fprintf(stdprn,"\tThe given raw data is :\n\n");
  for (i=0;i<nr;i++)
  {
    for (j=0;j<nsp;j++)
     fprintf(stdprn,"%f  ",x[i][j]);
    fprintf(stdprn,"\n");
  }
  fprintf(stdprn,"\n\n");


  correl(x,nsp,nr,r);
  flushall();

  fprintf(stdprn,"\n\n\tThe Correlation matrix is :\n\n");
  for (i=0;i<nsp;i++)
  {
    for (j=0;j<nsp;j++)
     fprintf(stdprn,"%f  ",r[i][j]);
    fprintf(stdprn,"\n");
  }
  fprintf(stdprn,"\n\n");

  if (mat_inv(r,nsp,r1)==0) goto out;


  fprintf(stdprn,"\n\n\tThe communalities are :\n\n");
  sum = 0.0;
  for (i=0;i<nsp;i++)
  {
    commun[i] = 1.0 - (1.0/r1[i][i]);
    sum = sum + commun[i];
    fprintf(stdprn,"\n\t%f\n",commun[i]);
  }
    fprintf(stdprn,"\n\n");

 for (i=0;i<nsp;i++)
   r[i][i] = commun[i];


 n1=factor(r,nsp,d,fact,var);
 if (n1 < 0) goto out;
 delay(2000);
 clrscr();


fprintf(stdprn,"The Eigenvalues are :\n\n");
for (i=0;i<nsp;i++)
  fprintf(stdprn,"\t%f\n",d[i][i]);
```

```
fprintf(stdprn,"\n\nFACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(stdprn,"%f  ",fact[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");


fprintf(stdprn,"\n\nVARIANCE: ");
for (j=0;j<nl;j++)
  fprintf(stdprn,"%f  ",var[j]);
fprintf(stdprn,"\n\n");

sum1 = 0.0;
fprintf(stdprn,"\n\nVARIANCE % : ");
for (j=0;j<nl;j++)
{
  sum1=sum1+(var[j]/sum)*100;
  fprintf(stdprn,"%6.2f  ",(var[j]/sum)*100);
}
fprintf(stdprn,"\n\nTotal variance accounted for : %6.2f\n\n",sum1);

clrscr();
gotoxy(5,10);
if(quartimax(fact,nsp,nl,b) < 0)goto out;
flushall();


fprintf(stdprn,"\n\nThe final solution after QUARTIMAX Rotation :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(stdprn,"%f  ",b[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");



exit(1);

out: fprintf(stdprn,"\n\nERROR...!!!");
  return(0);
}
```

```c
                /* QMAX6.C   */


#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "qrtmax.c"
#define  MAX  50

int main()
{
  int   nl=0,nsp,nr,i,j,il,jl;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],rl[MAX][MAX],var[MAX],
            fact[MAX][MAX],b[MAX][MAX],d[MAX][MAX],sum,suml;

  char  c,namel[12],name2[12];

  FILE  *fpl,*fp2;

  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F  A  C  T  O  R    A  N  A  L  Y  S  I  S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("Q U A R T I M A X - R O T A T I O N");
  normvideo();
  delay(2000);

  clrscr();
aal:  gotoxy(10,10);
  printf("Name of the data-file : ");
  gets(namel);
  flushall();


  fpl = fopen(namel,"r");
  if (!fpl)
  {
    gotoxy(10,15);
    highvideo();
    cprintf("No such file exists...!!!");
    normvideo();
    delay(500);
    gotoxy(10,15);
    clreol();
    gotoxy(25,10);
    clreol();
    goto aal;
  }


  clrscr();
aa:  gotoxy(10,10);
  printf("Name of the output-file : ");
  gets(name2);
  flushall();
```

```
 fp2 = fopen(name2,"r");
 if (fp2)
 {
   fclose(fp2);
   gotoxy(10,15);
   highvideo();
   cprintf("File already exists...!!!");
   normvideo();
   gotoxy(10,18);
   printf("Overwrite it...(Y/N)...??? ");
   c = getch();
   if (c == 'y' || c == 'Y')
     goto a1;
   gotoxy(10,18);
   clreol();
   gotoxy(10,15);
   clreol();
   gotoxy(25,10);
   clreol();
   goto aa;
 }



a1: clrscr();
  fp2 = fopen(name2,"w");
  if (!fp2)
  {
    fclose(fp2);
    gotoxy(10,15);
    highvideo();
    cprintf("Can't open the file...!!!");
    normvideo();
    delay(300);
    gotoxy(10,15);
    clreol();
    goto aa;
  }

  fscanf(fp1,"%d",&nsp);
  flushall();
  fscanf(fp1,"%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
    for (j=0;j<nsp;j++)
    {
      fscanf(fp1,"%f",&x[i][j]);
    }
  flushall();
  clrscr();

  gotoxy(10,5);
  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();

  fprintf(fp2,"\tThe given raw data is :\n\n");
  for (i=0;i<nr;i++)
  {
    for (j=0;j<nsp;j++)
     fprintf(fp2,"%f  ",x[i][j]);
    fprintf(fp2,"\n");
  }
```

```
fprintf(fp2,"\n\n");


correl(x,nsp,nr,r);
flushall();

fprintf(fp2,"\n\n\tThe Correlation matrix is :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nsp;j++)
   fprintf(fp2,"%f  ",r[i][j]);
  fprintf(fp2,"\n");
}
fprintf(fp2,"\n\n");

if (mat_inv(r,nsp,r1)==0) goto out;


fprintf(fp2,"\n\n\tThe communalities are :\n\n");
sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  fprintf(fp2,"\n\t%f\n",commun[i]);
}
  fprintf(fp2,"\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];


nl=factor(r,nsp,d,fact,var);
if (nl < 0) goto out;
delay(2000);
clrscr();


fprintf(fp2,"The Eigenvalues are :\n\n");
for (i=0;i<nsp;i++)
  fprintf(fp2,"\t%f\n",d[i][i]);




fprintf(fp2,"\n\nFACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(fp2,"%f  ",fact[i][j]);
  fprintf(fp2,"\n");
}
fprintf(fp2,"\n\n");


fprintf(fp2,"\n\nVARIANCE: ");
for (j=0;j<nl;j++)
  fprintf(fp2,"%f  ",var[j]);
fprintf(fp2,"\n\n");

sum1 = 0.0;
fprintf(fp2,"\n\nVARIANCE % : ");
for (j=0;j<nl;j++)
```

```
{
  suml=suml+(var[j]/sum)*100;
  fprintf(fp2,"%6.2f  ",(var[j]/sum)*100);
}
fprintf(fp2,"\n\nTotal variance accounted for : %6.2f\n\n",suml);


gotoxy(5,10);
if(quartimax(fact,nsp,nl,b) < 0)goto out;
flushall();
clrscr();

fprintf(fp2,"The final solution after QUARTIMAX Rotation :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(fp2,"%f  ",b[i][j]);
  fprintf(fp2,"\n");
}
fprintf(fp2,"\n\n");



exit(1);

out: fprintf(fp2,"\n\nERROR...!!!");
  return(0);
}
```

```
/* QRTMAX.C*/

/*   QUARTIMAX  */

#include <stdio.h>
#include <math.h>
#include <conio.h>
#define  MAX  50

int quartimax( float fact[MAX][MAX],int nsp,int nl,float b[MAX][MAX])
{
  int   i,j,il,jl,iteration;
  float a[MAX][MAX],t[MAX][MAX],
            sum4,tempsum,num,den,templ,temp2,phi,epslon = .00001;


  for (i=0;i<nsp;i++)
    for (j=0;j<nl;j++)
      a[i][j] = fact[i][j];

  sum4 = 0.0;
  for (i=0;i<nsp;i++)
    for (j=0;j<nl;j++)
      sum4 += (a[i][j]*a[i][j]*a[i][j]*a[i][j]);


  iteration = 0;
  do
  {
   gotoxy(1,wherey());
   printf("Iteration = %d sum = %f",iteration,sum4);
   delay(200);
   for (i=0;i<nl-1;i++)
   {
     for (j=i+1;j<nl;j++)
     {
          num = 0.0;
          den = 0.0;
          for (jl=0;jl<nsp;jl++)
          {
            templ = (2*a[jl][i]*a[jl][j]);
            temp2 = (a[jl][i]*a[jl][i]-a[jl][j]*a[jl][j]);
            num += templ*temp2;
            den += (temp2*temp2 - templ*templ);
          }
          num *= 2.0;
          phi = atan(num/den)/4.0 ;
          for (il=0;il<nl;il++)
            for (jl=0;jl<nl;jl++)
            {
             if (il==i && jl==i)
               t[il][jl] = cos(phi);
             else
               if (il==i && jl==j)
                      t[il][jl] = -sin(phi);
               else
                      if (il==j && jl==i)
                        t[il][jl] = sin(phi);
                      else
                        if (il==j && jl==j)
                          t[il][jl] = cos(phi);
```

```
                        else
                          if (il==jl)
                            t[il][jl] = 1.0;
                          else
                            t[il][jl] = 0.0;
                }
                if (mat_mul(a,nsp,nl,t,nl,nl,b) == 0)     goto out;
                for (i=0;i<nsp;i++)
                  for (j=0;j<nl;j++)
                    a[i][j] = b[i][j];
        }
      }


      tempsum = 0.0;
      for (i=0;i<nsp;i++)
        for (j=0;j<nl;j++)
                tempsum += (a[i][j]*a[i][j]*a[i][j]*a[i][j]);
      if (fabs(tempsum - sum4) < epslon)
      {
        gotoxy(1,wherey());
        printf("Iteration = %d sum = %f",iteration,sum4);
        return(1);
      }
      else
        sum4 = tempsum;
      iteration ++;
    }
    while (iteration < 5000);
    return(2);
  out: return(-1);
  }
```

```
/************ T R A N S P O S E . C *****************/

#define MAX 50

extern void transpose(float a[MAX][MAX],int n,int p,float b[MAX][MAX])
{
int i,j;
  for (i=0;i<p;i++)
     for (j=0;j<n;j++)
              b[i][j] = a[j][i];
}
```

```
/* VARIMAX.C*/

/*    VARIMAX  */


#include <stdio.h>
#include <math.h>
#include <conio.h>
#define  MAX  50

int varimax(float fact[MAX][MAX],int nsp,int nl,float commun[MAX],float b[MAX][MAX])
{

  int   i,j,il,jl,iteration;
  float a[MAX][MAX],t[MAX][MAX],sum11,sum21,sum1,sum2,sum3,sum4,uj,vj,phai,
            vsum,tempsum,num,den,temp1,temp2,phi,sum23,epslon = .00001;



  for (i=0;i<nsp;i++)
    for (j=0;j<nl;j++)
    {
      a[i][j] = fact[i][j]/sqrt(commun[i]);
    }


  vsum = 0.0;
  sum11 = sum23 = 0.0;
  for (j=0;j<nl;j++)
  {
    sum21 = 0.0;
    for (i=0;i<nsp;i++)
    {
      sum11 += (a[i][j]*a[i][j]*a[i][j]*a[i][j]);
      sum21 += (a[i][j]*a[i][j]);
    }
    sum21 *= sum21;
    sum23 += sum21;
  }
  vsum = (nsp*sum11) - sum23 ;


  iteration = 0;
  do
  {
    gotoxy(1,wherey());
    printf("Iteration = %d sum = %f",iteration,vsum);
    delay(200);
    for (i=0;i<nl-1;i++)
    {
      for (j=i+1;j<nl;j++)
      {
            sum1 = sum2 = sum3 = sum4 = 0.0 ;
            for (jl=0;jl<nsp;jl++)
            {
              uj = a[jl][i]*a[jl][i] - a[jl][j]*a[jl][j] ;
              vj = 2*a[jl][i]*a[jl][j] ;
              sum1 += uj;
              sum2 += vj;
              sum3 += (uj*uj - vj*vj);
              sum4 += (uj*vj);
            }
            sum4 *= 2.0;
```

```
            num = (sum4-((2*sum1*sum2)/nsp));
            den = (sum3-((sum1*sum1-sum2*sum2)/nsp));
            phai = (atan(num/den))/4.0;
            for (i1=0;i1<n1;i1++)
              for (j1=0;j1<n1;j1++)
              [
                if (i1==i && j1==i)
                  t[i1][j1] = cos(phai);
                else
                  if (i1==i && j1==j)
                         t[i1][j1] = -sin(phai);
                  else
                         if (i1==j && j1==i)
                           t[i1][j1] = sin(phai);
                         else
                           if (i1==j && j1==j)
                             t[i1][j1] = cos(phai);
                           else
                             if (i1==j1)
                               t[i1][j1] = 1.0;
                             else
                               t[i1][j1] = 0.0;
              |
            if (mat_mul(a,nsp,n1,t,n1,n1,b) == 0)    goto out;
            for (i=0;i<nsp;i++)
              for (j=0;j<n1;j++)
                a[i][j] = b[i][j];
      |
    |
    tempsum = sum11 = sum23 = 0.0;
    for (j1=0;j1<n1;j1++)
    [
      sum21 = 0.0;
      for (i1=0;i1<nsp;i1++)
      [
            sum11 += (a[i1][j1]*a[i1][j1]*a[i1][j1]*a[i1][j1]);
            sum21 += (a[i1][j1]*a[i1][j1]);
      |
      sum21 *= sum21;
      sum23 += sum21;
    |
    tempsum = (nsp*sum11) - sum23 ;


    if ((fabs(tempsum) - fabs(vsum)) < epslon)
      [
            gotoxy(1,wherey());
            printf("Iteration = %d sum = %f",iteration,tempsum);
            goto a1;
      |
    else vsum = tempsum;
    iteration ++;
  |
  while(iteration < 5000);

a1: for (i=0;i<nsp;i++)
      for (j=0;j<n1;j++)
            a[i][j] = a[i][j] * sqrt(commun[i]);


  return(1);
out: return(-1);
}
```

```
/* VARIMAX1.C */


#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "varimax.c"
#define MAX 50


int main()
{
  int   nl=0,nsp,nr,i,j,il,jl;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],rl[MAX][MAX],var[MAX],d[MAX][MAX],
            fact[MAX][MAX],b[MAX][MAX],sum,suml;



  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F A C T O R   A N A L Y S I S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("V A R I M A X - R O T A T I O N");
  normvideo();
  delay(2000);
  clrscr();


  gotoxy(10,5);
  printf("No. of variables ? : ");
  scanf("%d",&nsp);
  flushall();
  gotoxy(10,10);
  printf("No. of cases ? : ");
  scanf("%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
    for (j=0;j<nsp;j++)
    {
      gotoxy(10,15);
      printf("Input value for case %d variable %d : ",i+1,j+1);
      scanf("%f",&x[i][j]);
      gotoxy(22,15);
      clreol();
    }
  flushall();
  fcloseall();
  clrscr();

  gotoxy(10,5);
  printf("Press any key and you see the raw data...!!!");
  getch();
  flushall();
  printf("\n\n");
```

```
for (i=0;i<nr;i++)
{
  for (j=0;j<nsp;j++)
   printf("%f  ",x[i][j]);
  printf("\n");
}
printf("\n\n");
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();


correl(x,nsp,nr,r);
gotoxy(10,5);
printf("!!!...Press any key again to see the correlation matrix...!!!");
getch();
flushall();
printf("\n\n");

for (i=0;i<nsp;i++)
{
  for (j=0;j<nsp;j++)
   printf("%f  ",r[i][j]);
  printf("\n");
}
printf("\n\n");
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();

if (mat_inv(r,nsp,r1)==0) goto out;

printf("/...Press a key to see the communalities found by SMC method...\\");
getch();
flushall();
printf("\n\n");

sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  printf("\n\t%f\n",commun[i]);
}
printf("\n\n");
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();
gotoxy(10,15);
highvideo();
cprintf("PLEASE WAIT....EIGENVALUES ARE COMING NEXT...");
normvideo();
printf("\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];


nl=factor(r,nsp,d,fact,var);
```

```
if (nl < 0) goto out;
delay(2000);
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();

gotoxy(10,10);
printf("->>>Press a key and get the Eigenvalues ->>>\n\n");
getch();

flushall();
for (i=0;i<nsp;i++)
  printf("\t%f\n",d[i][i]);
printf("\nObviously you have to PRESS a key to get...");
getch();
flushall();
clrscr();



printf("FACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    printf("%f  ",fact[i][j]);
  printf("\n");
}
printf("\n\n");



printf("VARIANCE: ");
for (j=0;j<nl;j++)
  printf("%f  ",var[j]);
printf("\n\n");

suml = 0.0;
printf("VARIANCE % : ");
for (j=0;j<nl;j++)
{
  suml=suml+(var[j]/sum)*100;
  printf("%6.2f  ",(var[j]/sum)*100);
}
printf("\n\nTotal variance accounted for : %6.2f\n\n",suml);

printf("\n\nPress a key to get the final solution using VARIMAX method\n\n");
getch();
flushall();
clrscr();

gotoxy(5,10);
if(varimax(fact,nsp,nl,commun,b) < 0)goto out;
flushall();
clrscr();

printf("The final solution after VARIMAX Rotation :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    printf("%f  ",b[i][j]);
  printf("\n");
}
```

```
  printf("\n\n");

  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();
  clrscr();
  exit(1);

out: printf("\n\nERROR...!!!");
 return(0);
}
```

```
fprintf(stdprn,"\n\n\tThe give raw data....\n\n");
for (i=0;i<nr;i++)
{
  for (j=0;j<nsp;j++)
   fprintf(stdprn,"%f  ",x[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");
clrscr();

correl(x,nsp,nr,r);
gotoxy(10,5);
fprintf(stdprn,"The correlation matrix is :\n\n");

for (i=0;i<nsp;i++)
{
  for (j=0;j<nsp;j++)
   fprintf(stdprn,"%f  ",r[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");

if (mat_inv(r,nsp,r1)==0) goto out;

fprintf(stdprn,"The communalities using SMC method are ....\n\n");

sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  fprintf(stdprn,"\n\t%f\n",commun[i]);
}
fprintf(stdprn,"\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];



nl=factor(r,nsp,d,fact,var);
if (nl < 0) goto out;
delay(2000);
clrscr();

gotoxy(10,10);
fprintf(stdprn,"\nThe Eigenvalues are :\n\n");
for (i=0;i<nsp;i++)
  fprintf(stdprn,"\t%f\n",d[i][i]);


fprintf(stdprn,"FACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(stdprn,"%f  ",fact[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");



fprintf(stdprn,"VARIANCE: ");
```

```
    for (j=0;j<nsp;j++)
     fprintf(fp1,"%f  ",r[i][j]);
    fprintf(fp1,"\n");
  }
  fprintf(fp1,"\n\n");

  if (mat_inv(r,nsp,r1)==0) goto out;

  fprintf(fp1,"The communalities using SMC method are ....\n\n");

  sum = 0.0;
  for (i=0;i<nsp;i++)
  {
    commun[i] = 1.0 - (1.0/r1[i][i]);
    sum = sum + commun[i];
    fprintf(fp1,"\n\t%f\n",commun[i]);
  }
  fprintf(fp1,"\n\n");

  for (i=0;i<nsp;i++)
    r[i][i] = commun[i];



  n1=factor(r,nsp,d,fact,var);
  if (n1 < 0) goto out;
  delay(2000);
  clrscr();

  gotoxy(10,10);
  fprintf(fp1,"\nThe Eigenvalues are :\n\n");
  for (i=0;i<nsp;i++)
    fprintf(fp1,"\t%f\n",d[i][i]);


  fprintf(fp1,"FACTOR LOADINGS:\n\n");
  for (i=0;i<nsp;i++)
  {
    for (j=0;j<n1;j++)
      fprintf(fp1,"%f  ",fact[i][j]);
    fprintf(fp1,"\n");
  }
  fprintf(fp1,"\n\n");



  fprintf(fp1,"VARIANCE: ");
  for (j=0;j<n1;j++)
    fprintf(fp1,"%f  ",var[j]);
  fprintf(fp1,"\n\n");

  sum1 = 0.0;
  fprintf(fp1,"VARIANCE % : ");
  for (j=0;j<n1;j++)
  {
    sum1=sum1+(var[j]/sum)*100;
    fprintf(fp1,"%6.2f  ",(var[j]/sum)*100);
  }
  fprintf(fp1,"\n\nTotal variance accounted for : %6.2f\n\n",sum1);


  gotoxy(5,10);
  if(varimax(fact,nsp,n1,commun,b) < 0)goto out;
```

```
for (j=0;j<nl;j++)
  fprintf(stdprn,"%f  ",var[j]);
fprintf(stdprn,"\n\n");

sum1 = 0.0;
fprintf(stdprn,"VARIANCE % : ");
for (j=0;j<nl;j++)
{
  sum1=sum1+(var[j]/sum)*100;
  fprintf(stdprn,"%6.2f  ",(var[j]/sum)*100);
}
fprintf(stdprn,"\n\nTotal variance accounted for : %6.2f\n\n",sum1);

gotoxy(5,10);
if(varimax(fact,nsp,nl,commun,b) < 0)goto out;
flushall();
clrscr();

fprintf(stdprn,"\n\nThe final solution after VARIMAX Rotation :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(stdprn,"%f  ",b[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");



exit(1);

out: fprintf(stdprn,"\n\nERROR...!!!");
  return(0);
}
```

```
    clreol();
    goto aa;
  }

al: clrscr();
  fp1 = fopen(name,"w");
  if (!fp1)
  {
    fclose(fp1);
    gotoxy(10,15);
    highvideo();
    cprintf("Can't open the file...!!!");
    normvideo();
    delay(300);
    gotoxy(10,15);
    clreol();
    goto al;
  }
  gotoxy(10,5);
  printf("No. of variables ? : ");
  scanf("%d",&nsp);
  flushall();
  gotoxy(10,10);
  printf("No. of cases ? : ");
  scanf("%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
    for (j=0;j<nsp;j++)
    {
      gotoxy(10,15);
      printf("Input value for case %d variable %d : ",i+1,j+1);
      scanf("%f",&x[i][j]);
      gotoxy(22,15);
      clreol();
    }
  flushall();
  clrscr();

  gotoxy(10,5);
  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();


  fprintf(fp1,"\n\n\tThe give raw data....\n\n");
  for (i=0;i<nr;i++)
  {
    for (j=0;j<nsp;j++)
     fprintf(fp1,"%f  ",x[i][j]);
    fprintf(fp1,"\n");
  }
  fprintf(fp1,"\n\n");


  correl(x,nsp,nr,r);
  gotoxy(10,5);
  flushall();
  fprintf(fp1,"The correlation matrix is :\n\n");

  for (i=0;i<nsp;i++)
  {
```

```
 flushall();
 clrscr();

 fprintf(fp1,"\n\nThe final solution after VARIMAX Rotation :\n\n");
 for (i=0;i<nsp;i++)
 {
   for (j=0;j<nl;j++)
     fprintf(fp1,"%f  ",b[i][j]);
   fprintf(fp1,"\n");
 }
 fprintf(fp1,"\n\n");


 exit(1);

out: fprintf(fp1,"\n\nERROR...!!!");
  return(0);
}
```

```c
/* VARIMAX2.C  */



#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "varimax.c"
#define  MAX  50


int main()
{
  int   n1=0,nsp,nr,i,j,i1,j1;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],r1[MAX][MAX],var[MAX],
            fact[MAX][MAX],b[MAX][MAX],d[MAX][MAX],sum,sum1;



  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F  A  C  T  O  R    A  N  A  L  Y  S  I  S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("V A R I M A X - R O T A T I O N");
  normvideo();
  delay(2000);
  clrscr();


  gotoxy(10,5);
  printf("No. of variables ? : ");
  scanf("%d",&nsp);
  flushall();
  gotoxy(10,10);
  printf("No. of cases ? : ");
  scanf("%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
    for (j=0;j<nsp;j++)
    {
      gotoxy(10,15);
      printf("Input value for case %d variable %d : ",i+1,j+1);
      scanf("%f",&x[i][j]);
      gotoxy(22,15);
      clreol();
    }
 flushall();
 fcloseall();
 clrscr();

 gotoxy(10,5);
 printf("PRESS ANY KEY WHEN THE PRINTER IS READY...!!!");
 getch();
 flushall();
```

```
/* VARIMAX3.C  */



#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "varimax.c"
#define MAX 50


int main()
{
  int   nl=0,nsp,nr,i,j,il,jl;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],rl[MAX][MAX
             fact[MAX][MAX],d[MAX][MAX],b[MAX][MAX],sum,suml;

  char        c,name[12];

  FILE        *fpl;


  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F A C T O R   A N A L Y S I S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R  M E T H O D");
  gotoxy(20,17);
  cprintf("V A R I M A X - R O T A T I O N");
  normvideo();
  delay(2000);

  clrscr();
aa:  gotoxy(10,10);
  printf("Name of the output-file : ");
  gets(name);
  flushall();


  fpl = fopen(name,"r");
  if (fpl)
  {
    fclose(fpl);
    gotoxy(10,15);
    highvideo();
    cprintf("File already exists...!!!");
    normvideo();
    gotoxy(10,18);
    printf("Overwrite it...(Y/N)...??? ");
    c = getch();
    if (c == 'y' || c == 'Y')
      goto al;
    gotoxy(10,18);
    clreol();
    gotoxy(10,15);
    clreol();
    gotoxy(25,10);
```

```
/* VARIMAX4.C */

#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "varimax.c"
#define MAX 50

int main()
{
  int   n1=0,nsp,nr,i,j,i1,j1;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],r1[MAX][MAX],var[MAX],
            fact[MAX][MAX],d[MAX][MAX],b[MAX][MAX],sum,sum1;

  char  name[12];

  FILE  *fp1;

  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F  A  C  T  O  R     A  N  A  L  Y  S  I  S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R   M E T H O D");
  gotoxy(20,17);
  cprintf("V A R I M A X - R O T A T I O N");
  normvideo();
  delay(2000);

  clrscr();
aa:  gotoxy(10,10);
  printf("Name of the data-file : ");
  gets(name);
  flushall();


  fp1 = fopen(name,"r");
  if (!fp1)
  {
    gotoxy(10,15);
    highvideo();
    cprintf("No such file exists...!!!");
    normvideo();
    delay(500);
    gotoxy(10,15);
    clreol();
    gotoxy(25,10);
    clreol();
    goto aa;
  }


 fscanf(fp1,"%d",&nsp);
 flushall();
 fscanf(fp1,"%d",&nr);
 flushall();
```

```
for (i=0;i<nr;i++)
  for (j=0;j<nsp;j++)
  {
    fscanf(fp1,"%f",&x[i][j]);
  }
flushall();
fcloseall();
clrscr();

gotoxy(10,5);
printf("Press any key and you see the raw data...!!!");
getch();
flushall();
printf("\n\n");

for (i=0;i<nr;i++)
{
  for (j=0;j<nsp;j++)
   printf("%f  ",x[i][j]);
  printf("\n");
}
printf("\n\n");
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();


correl(x,nsp,nr,r);
flushall();
gotoxy(10,5);
printf("!!!...Press any key again to see the correlation matrix...!!!");
getch();
flushall();
printf("\n\n");

for (i=0;i<nsp;i++)
{
  for (j=0;j<nsp;j++)
   printf("%f  ",r[i][j]);
  printf("\n");
}
printf("\n\n");
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();

if (mat_inv(r,nsp,r1)==0) goto out;

printf("/...Press a key to see the communalities found by SMC method...\\");
getch();
flushall();
printf("\n\n");

sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  printf("\n\t%f\n",commun[i]);
}
  printf("\n\n");
```

```
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();
gotoxy(10,15);
highvideo();
cprintf("PLEASE WAIT....EIGENVALUES ARE COMING NEXT...");
normvideo();
printf("\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];


nl=factor(r,nsp,d,fact,var);
if (nl < 0) goto out;
delay(2000);
printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();

gotoxy(10,10);
printf("->>>Press a key and get the Eigenvalues ->>>\n\n");
getch();
flushall();
for (i=0;i<nsp;i++)
  printf("\t%f\n",d[i][i]);
printf("\nObviously you have to PRESS a key to get...");
getch();
flushall();
clrscr();




printf("FACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    printf("%f  ",fact[i][j]);
  printf("\n");
}
printf("\n\n");


printf("VARIANCE: ");
for (j=0;j<nl;j++)
  printf("%f  ",var[j]);
printf("\n\n");

suml = 0.0;
printf("VARIANCE % : ");
for (j=0;j<nl;j++)
{
  suml=suml+(var[j]/sum)*100;
  printf("%6.2f  ",(var[j]/sum)*100);
}
printf("\n\nTotal variance accounted for : %6.2f\n\n",suml);


printf("\n\nPress a key to get the final solution using VARIMAX method\n\n");
getch();
```

```
flushall();
clrscr();

gotoxy(5,10);
if(varimax(fact,nsp,nl,commun,b) < 0)goto out;
flushall();
clrscr();

printf("The final solution after VARIMAX Rotation :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    printf("%f  ",b[i][j]);
  printf("\n");
}
printf("\n\n");


printf("PRESS ANY KEY...!!!");
getch();
flushall();
clrscr();


exit(1);

out: printf("\n\nERROR...!!!");
 return(0);
}
```

```
/* VARIMAX5.C  */

#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "varimax.c"
#define MAX 50

int main()
{
  int  nl=0,nsp,nr,i,j,il,jl;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],rl[MAX][MAX],var[MAX],
            fact[MAX][MAX],b[MAX][MAX],d[MAX][MAX],sum,suml;

  char  name[12];

  FILE  *fpl;

  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F  A  C  T  O  R    A  N  A  L  Y  S  I  S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R   M E T H O D");
  gotoxy(20,17);
  cprintf("V A R I M A X - R O T A T I O N");
  normvideo();
  delay(2000);

  clrscr();
aa:  gotoxy(10,10);
  printf("Name of the data-file : ");
  gets(name);
  flushall();


  fpl = fopen(name,"r");
  if (!fpl)
  {
    gotoxy(10,15);
    highvideo();
    cprintf("No such file exists...!!!");
    normvideo();
    delay(500);
    gotoxy(10,15);
    clreol();
    gotoxy(25,10);
    clreol();
    goto aa;
  }


  fscanf(fpl,"%d",&nsp);
  flushall();
  fscanf(fpl,"%d",&nr);
```

```c
flushall();
for (i=0;i<nr;i++)
  for (j=0;j<nsp;j++)
  {
    fscanf(fp1,"%f",&x[i][j]);
  }
flushall();
clrscr();

gotoxy(10,5);
printf("PRESS ANY KEY WHEN THE PRINTER IS READY...!!!");
getch();
flushall();

fprintf(stdprn,"\tThe given raw data is :\n\n");
for (i=0;i<nr;i++)
{
  for (j=0;j<nsp;j++)
   fprintf(stdprn,"%f  ",x[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");


correl(x,nsp,nr,r);
flushall();

fprintf(stdprn,"\n\n\tThe Correlation matrix is :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nsp;j++)  .
   fprintf(stdprn,"%f  ",r[i][j]);
  fprintf(stdprn,"\n");
}
fprintf(stdprn,"\n\n");

if (mat_inv(r,nsp,r1)==0) goto out;


fprintf(stdprn,"\n\n\tThe communalities are :\n\n");
sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  fprintf(stdprn,"\n\t%f\n",commun[i]);
}
  fprintf(stdprn,"\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];


n1=factor(r,nsp,d,fact,var);
if (n1 < 0) goto out;
delay(2000);
clrscr();


fprintf(stdprn,"The Eigenvalues are :\n\n");
for (i=0;i<nsp;i++)
  fprintf(stdprn,"\t%f\n",d[i][i]);
```

```
  fprintf(stdprn,"\n\nFACTOR LOADINGS:\n\n");
  for (i=0;i<nsp;i++)
  {
    for (j=0;j<nl;j++)
      fprintf(stdprn,"%f  ",fact[i][j]);
    fprintf(stdprn,"\n");
  }
  fprintf(stdprn,"\n\n");


  fprintf(stdprn,"\n\nVARIANCE: ");
  for (j=0;j<nl;j++)
    fprintf(stdprn,"%f  ",var[j]);
  fprintf(stdprn,"\n\n");

  sum1 = 0.0;
  fprintf(stdprn,"\n\nVARIANCE % : ");
  for (j=0;j<nl;j++)
  {
    sum1=sum1+(var[j]/sum)*100;
    fprintf(stdprn,"%6.2f  ",(var[j]/sum)*100);
  }
  fprintf(stdprn,"\n\nTotal variance accounted for : %6.2f\n\n",sum1);

  gotoxy(5,10);
  if(varimax(fact,nsp,nl,commun,b) < 0)goto out;
  flushall();
  clrscr();

  fprintf(stdprn,"\n\nThe final solution after VARIMAX Rotation :\n\n");
  for (i=0;i<nsp;i++)
  {
    for (j=0;j<nl;j++)
      fprintf(stdprn,"%f  ",b[i][j]);
    fprintf(stdprn,"\n");
  }
  fprintf(stdprn,"\n\n");




  exit(1);

out: fprintf(stdprn,"\n\nERROR...!!!");
  return(0);
}
```

```
                    /* VARIMAX6.C    */



#include <stdio.h>
#include <math.h>
#include <conio.h>
#include "correl.c"
#include "mat_inv.c"
#include "factor.c"
#include "varimax.c"
#define  MAX  50

int main()
{
  int   nl=0,nsp,nr,i,j,il,jl;

  float x[MAX][MAX],commun[MAX],r[MAX][MAX],rl[MAX][MAX],var[MAX],
            fact[MAX][MAX],b[MAX][MAX],d[MAX][MAX],sum,suml;

  char  c,name1[12],name2[12];

  FILE  *fp1,*fp2;

  clrscr();
  gotoxy(15,2);
  highvideo();
  cprintf("F  A  C  T  O  R    A  N  A  L  Y  S  I  S");
  gotoxy(15,10);
  cprintf("P R I N C I P A L - F A C T O R   M E T H O D");
  gotoxy(20,17);
  cprintf("V A R I M A X - R O T A T I O N");
  normvideo();
  delay(2000);

  clrscr();
aal:  gotoxy(10,10);
  printf("Name of the data-file : ");
  gets(name1);
  flushall();


  fp1 = fopen(name1,"r");
  if (!fp1)
  {
    gotoxy(10,15);
    highvideo();
    cprintf("No such file exists...!!!");
    normvideo();
    delay(500);
    gotoxy(10,15);
    clreol();
    gotoxy(25,10);
    clreol();
    goto aal;
  }


  clrscr();
aa:  gotoxy(10,10);
```

```
  printf("Name of the output-file : ");
  gets(name2);
  flushall();


  fp2 = fopen(name2,"r");
  if (fp2)
  {
    fclose(fp2);
    gotoxy(10,15);
    highvideo();
    cprintf("File already exists...!!!");
    normvideo();
    gotoxy(10,18);
    printf("Overwrite it...(Y/N)...??? ");
    c = getch();
    if (c == 'y' || c == 'Y')
      goto al;
    gotoxy(10,18);
    clreol();
    gotoxy(10,15);
    clreol();
    gotoxy(25,10);
    clreol();
    goto aa;
  }



al: clrscr();
  fp2 = fopen(name2,"w");
  if (!fp2)
  {
    fclose(fp2);
    gotoxy(10,15);
    highvideo();
    cprintf("Can't open the file...!!!");
    normvideo();
    delay(300);
    gotoxy(10,15);
    clreol();
    goto al;
  }

  fscanf(fp1,"%d",&nsp);
  flushall();
  fscanf(fp1,"%d",&nr);
  flushall();
  for (i=0;i<nr;i++)
    for (j=0;j<nsp;j++)
    {
      fscanf(fp1,"%f",&x[i][j]);
    }
  flushall();
  clrscr();

  gotoxy(10,5);
  printf("PRESS ANY KEY...!!!");
  getch();
  flushall();

  fprintf(fp2,"\tThe given raw data is :\n\n");
  for (i=0;i<nr;i++)
```

```
{
  for (j=0;j<nsp;j++)
   fprintf(fp2,"%f  ",x[i][j]);
  fprintf(fp2,"\n");
}
fprintf(fp2,"\n\n");


correl(x,nsp,nr,r);
flushall();

fprintf(fp2,"\n\n\tThe Correlation matrix is :\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nsp;j++)
   fprintf(fp2,"%f  ",r[i][j]);
  fprintf(fp2,"\n");
}
fprintf(fp2,"\n\n");

if (mat_inv(r,nsp,r1)==0) goto out;


fprintf(fp2,"\n\n\tThe communalities are :\n\n");
sum = 0.0;
for (i=0;i<nsp;i++)
{
  commun[i] = 1.0 - (1.0/r1[i][i]);
  sum = sum + commun[i];
  fprintf(fp2,"\n\t%f\n",commun[i]);
}
  fprintf(fp2,"\n\n");

for (i=0;i<nsp;i++)
  r[i][i] = commun[i];


nl=factor(r,nsp,d,fact,var);
if (nl < 0) goto out;
delay(2000);
clrscr();


fprintf(fp2,"The Eigenvalues are :\n\n");
for (i=0;i<nsp;i++)
  fprintf(fp2,"\t%f\n",d[i][i]);




fprintf(fp2,"\n\nFACTOR LOADINGS:\n\n");
for (i=0;i<nsp;i++)
{
  for (j=0;j<nl;j++)
    fprintf(fp2,"%f  ",fact[i][j]);
  fprintf(fp2,"\n");
}
fprintf(fp2,"\n\n");


fprintf(fp2,"\n\nVARIANCE: ");
for (j=0;j<nl;j++)
  fprintf(fp2,"%f  ",var[j]);
```

```
 fprintf(fp2,"\n\n");

 suml = 0.0;
 fprintf(fp2,"\n\nVARIANCE % : ");
 for (j=0;j<nl;j++)
 {
   suml=suml+(var[j]/sum)*100;
   fprintf(fp2,"%6.2f  ",(var[j]/sum)*100);
 }
 fprintf(fp2,"\n\nTotal variance accounted for : %6.2f\n\n",suml);


 gotoxy(5,10);
 if(varimax(fact,nsp,nl,commun,b) < 0)goto out;
 flushall();
 clrscr();

 fprintf(fp2,"\n\nThe final solution after VARIMAX Rotation :\n\n");
 for (i=0;i<nsp;i++)
 {
   for (j=0;j<nl;j++)
     fprintf(fp2,"%f  ",b[i][j]);
   fprintf(fp2,"\n");
 }
 fprintf(fp2,"\n\n");



 exit(1);

out: fprintf(fp2,"\n\nERROR...!!!");
  return(0);
}
```

APPENDIX B

References

Appendix B

References

1. Annual Report 1991, TERI.
2. Mukhi, C++
3. John, A. Cooper and John, G. Watson, Jr., (copyright 1980, Air Pollution Control Association)
4. Ronald C. Henry *et.al*, Review of Receptor Model Fundamentals. (A report from the Mathematical and Empirical Receptor Models Workshop (Quail and Roost II) - 1984)
5. Introductory Computer Methods and Numerical Analysis.(ref: 517.949:681.3, IIT Library, Delhi)
6. Harry H. Harman, Modern Factor Analysis, The University of Chicago Press, Chicago and London, 1970.
7. Manuals, Borland Turbo C++, (copyright 1990).